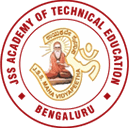
**JSS Mahavidyapeetha**

**JSS Academy of Technical Education**

**Kengeri - Uttarahalli Main Road, Bangalore-560060**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**ASSIGNMENT PROJECT SYNOPSIS**

**COURSE NAME: PYTHON APPLICATION PROGRAMMING**

**COURSE CODE: 17CS664**

**TOPIC: SUSPESION DETECTION AND REPORTING BOT**

**UNDER THE GUIDENCE OF:**

**Dr NAVEEN NC**

**Prepared by: - HARSHITH R SHEKAR[1JS17CS039]**

**DHRUVA V [1JS18CS403]**

**BINDUSHREE R [1JS17CS025]**

**Marks Scored: Semester / Branch: 6th ‘A’ sec**

**Signature of the Staff:**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **NAME** | **Page Number** |
| [Overview](#overview) | 2 |
| [Background Work](#background) | 3 |
| [Architecture](#architecture) | 4 |
| [Implementation](#implementation) | 6 |
| [Area of Focus / Optimizations](#Area) | 30 |
| [Results / Conclusion](#result) | 31 |
| [References](#ref) | 35 |

**OVERVIEW**

The ATM System is the project which is used to access their bank accounts to make cash withdrawals and deposits. Whenever the user needs to make cash withdraws, they can enter their PIN number (personal identification number) and it will display the amount to be withdrawn and deposit. Once their withdrawn was successful, the amount will be debited in their account and if deposit is complete, amount will be credited to the account.

The ATM System is developed in Python and a CSV file (Comma-separated values) using file read write. Python is relatively simple, so it is easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers. Hence, we use this software in our project.

The ATM will service one customer at a time. A customer will be required to enter ATM Card number, personal identification number (PIN) – both of 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. Also, customer must be able to make a balance inquiry of any account linked to the card.

If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.

Millions of times per day around the globe people are instantly withdrawing money at automatic teller machines (ATMs). Given the fast pace of the world today, it is not surprising that the demand for access to quick cash is so immense. The power of ATMs would not be possible without secure connections. The final act of ATM dispending cash is the result of an amazingly fast burst of the customer never sees, but a trust is being done in a confidential manner.

**BACKGROUND WORK**

PYTHON LIBRARIES

Since some print statements can be parsed as function calls or statements, python 2 to python 3 cannot always read files containing the print function. When 2 to 3 detects the presence of the from \_\_future\_\_ import print\_function compiler directive, it modifies its internal grammar to interpret print() as a function. This change can also be enabled manually with the -p flag. Use -p to run fixers on code that already has had its print statements converted.

os module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

sys module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.

The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon. For details of SMTP and ESMTP operation, consult RFC 821 (Simple Mail Transfer Protocol) and RFC 1869 (SMTP Service Extensions).

Time module provides various time-related functions

The datetime module supplies classes for manipulating dates and times.

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. CSV format was used for many years prior to attempts to describe the format in a standardized way in RFC 4180. The lack of a well-defined standard means that subtle differences often exist in the data produced and consumed by different applications. These differences can make it annoying to process CSV files from multiple sources. Still, while the delimiters and quoting characters vary, the overall format is similar enough that it is possible to write a single module which can efficiently manipulate such data, hiding the details of reading and writing the data from the programmer.

re module provides regular expression matching operations similar to those found in Perl.

random module implements pseudo-random number generators for various distributions.

The getpass module provides two functions:

* getpass.getpass(prompt='Password: ', stream=None)
* getpass.getuser()

**ARCHITECTURE**

ATM Functionalities(Using Basic Concepts of Python):

1. Account Handling

2. Activation\De-Activation of Accounts

3. Admin Control

4. File Handling (using csv file)

5. Account Number Auto-Generation

6. Pin Auto-Generation

7. Simple Encryption\Decryption

8. Date and Time Implication

9. Exceptional Handling

10. Email Notifier Using SMTP Library

11. Amount Transfer

12. Made Applicable for Any Version of Python

Then the functionalities are divided by different python files,

* **Python File** : acc\_no\_gen.py

Functionalities implemented Account Number Auto-Generation , Exceptional Handling and Made Applicable for Any Version of Python.

* **Python File** : ATM.py

Functionalities implemented Account Handling, Amount Transfer, Exceptional Handling and Made Applicable for Any Version of Python.

* **Python File** : Data.py

Functionalities implemented File Handling (using csv file), Exceptional Handling and Made Applicable for Any Version of Python.

* **Python File** : encrypt.py

Functionalities implemented Simple Encryption\Decryption.

* **Python File** : login.py

Functionalities implemented Admin Control, Pin Auto-Generation, Date and Time Implication, Exceptional Handling and Made Applicable for Any Version of Python.

* **Python File** : acc\_no\_gen.py

Functionalities implemented Email Notifier Using SMTP Library, Exceptional Handling and Made Applicable for Any Version of Python.

**Implementation**

* **Python File** : acc\_no\_gen.py

from random import randint, randrange

from Data import data

def account\_no\_gen(user\_name):

d = data()

alphabets = 'abcdefghijklmnopqrstuvwxyz'

acc\_no = ''

acc\_no += str(len(d.keys()) + 10)

for alpha in user\_name:

if (len(acc\_no) < 12):

if alpha in alphabets:

index = alphabets.rfind(alpha)

acc\_no += str(index + 1)

else:

acc\_no += '0'

else:

break

if len(acc\_no) > 12:

final\_acc\_no = ''

for index in acc\_no:

if len(final\_acc\_no) < 12:

final\_acc\_no += index

acc\_no = final\_acc\_no

return acc\_no

if len(acc\_no) < 12:

remain\_index = 12 - len(acc\_no)

for index in range(remain\_index):

acc\_no += str(randint(0,9))

return acc\_no

else:

return acc\_no

def code():

a = 'qwertyuiopasdfghjklzxcvbnm'

conf\_code = ''

for i in range(3):

conf\_code += str(a[randrange(9)])+str(randrange(9))

return conf\_code

* **Python File** : ATM.py

#IMPORTS

## USING PYTHON BUILT-IN LIBRARIES

from \_\_future\_\_ import print\_function

from getpass import getpass as gp

import os, csv

##USING SOURCE FILES

from encrypt import rot13

from Data import join,data

from send\_mail import sendmail

from acc\_no\_gen import code

#END OF IMPORTS

#Counter for user amount

net\_balance = 0.0

# Using input() in python 2 or 3

try:

# set raw\_input as input in python2

input = raw\_input

except:

pass

#Atm function called after successfull login

def atm(user\_name,Net\_balance,Pin,History,acc\_no,address):

filename = join()

clear = ('cls' if os.name == 'nt' else 'clear')

#input for change of pin

# new\_pin\_opt = input("Change Pin : \n1. Yes \n2. No \n")

# os.system(clear)

# if (new\_pin\_opt == '1') or (new\_pin\_opt.lower().startswith('y')):

import time,datetime

print (time.strftime('Date:%d-%b-%Y \nTime:%I:%M %p Today:%A\n'))

print ("""

Y Y 000 BBBBBB

Y Y 00 00 B B

Y Y 00 00 B B

Y 00 00 BBBBBB

Y 00 00 B B

Y 00 00 B B

Y 000 BBBBBB

""")

print(("DEAR"),(user\_name.upper())+("!"))

print("WELCOME TO YOB SERVICE \n")

#User input for selection

global net\_balance

net\_balance += Net\_balance

Opr = input(":: Please Select An Option Provided Below : \n1. Check Account Balance \n2. Check Acount Number \n3. Deposit \n4. Withdraw \n5. Transfer Amount \n6. Last Acive Session \n7. Change Pin \n8. Change/Verify Mail Address \n0. Exit \n")

os.system(clear)

if not Opr.isdigit():

Opr = 9

while int(Opr) != 0:

if int(Opr) == 1:

os.system(clear)

print (":: Your Acount Balance = Rs","{:,} ::".format(net\_balance),"\n")

elif int(Opr) == 2:

os.system(clear)

print(":: Your Account Number =",acc\_no,":: \n")

#Deposit function is called

elif int(Opr) == 3:

os.system(clear)

deposit(net\_balance, address)

#Withdraw function is called

elif int(Opr) == 4:

os.system(clear)

withdraw(net\_balance, address)

#Amount Transfer function is called

elif int(Opr) == 5:

os.system(clear)

if net\_balance < 0.0:

print (":: Amount Can Not Be Transferred! ::\n:: Your Acount Balance = Rs",balance,"::","\n")

else:

account\_no = input('Enter 12-Digit Account Number : ')

if (account\_no == acc\_no):

os.system(clear)

print(":: Amount Transfer Not Possible! ::")

print(":: Provided Account Number Is Yours! ::\n")

else:

amount = amount\_transfer(account\_no, net\_balance, acc\_no, address)

net\_balance -= float(amount)

elif int(Opr) == 6:

os.system(clear)

print (":: Your Acount Was Previously Logged in on",History,"::","\n")

#Change Pin function is called

elif int(Opr) == 7:

os.system(clear)

Pin = change\_pin(Pin, address)

elif int(Opr) == 8:

os.system(clear)

Mail\_address = input(":: Please Enter A Valid Email Address : ")

conf = code()

MSG = "Confirming Mail Address.\n\nAccount Number : "+acc\_no+"\n\nYour Verification Code Is : '"+conf+"'"

if Mail\_address == address:

os.system(clear)

print(":: Your Email Address Is Verified! ::\n:: Want To Change Your Address ?")

opt = input("1. Yes\n2. No \n")

os.system(clear)

if opt.lower().startswith("y") or opt == '1':

Mail\_address = input(":: Please Enter A New Valid Email Address : ")

if Mail\_address == address:

os.system(clear)

print(":: Email Address Already Verified! ::")

else:

verify = sendmail(Mail\_address, MSG, "Confirmation Mail")

if (verify == True):

os.system(clear)

print("Verification Code Has Been Sent To Your Email Address!")

user\_conf = input("Enter Provided Code : ")

if (user\_conf.lower() == conf.lower()):

address = Mail\_address

os.system(clear)

print("Email Address Verified And Changed Successfully!\n")

else:

os.system(clear)

print("Invalid Code!\nYour Email Could Not Be Verified.\nYou May Try Again Later!\n")

else:

os.system(clear)

print(verify)

else:

print(":: Email Address Unchanged! ::")

else:

os.system(clear)

verify = sendmail(Mail\_address, MSG, "Confirmation Mail")

if (verify == True):

os.system(clear)

print("Verification Code Has Been Sent To Your Email Address!")

user\_conf = input("Enter Provided Code : ")

if (user\_conf.lower() == conf.lower()):

address = Mail\_address

os.system(clear)

print("Email Address Verified And Changed Successfully\n")

else:

os.system(clear)

print("Invalid Code!\nYour Email Could Not Be Verified.\nYou May Try Again Later!\n")

else:

os.system(clear)

print(verify)

else:

os.system(clear)

print (":: Invalid Selection! ::")

#Incase above condition(s) get meet

#Loop continues untill '0' is entered

Opr = input(":: Please Select An Option Provided Below : \n1. Check Account Balance \n2. Check Acount Number \n3. Deposit \n4. Withdraw \n5. Transfer Amount \n6. Last Acive Session \n7. Change Pin \n8. Change/Verify Mail Address \n0. Exit \n")

if not Opr.isdigit():

Opr = 9

os.system(clear)

os.system(clear)

print ("::: Thanks For Using ATM! :::\n::: We Hope You Are Satisfied With Our Service. :::\n::: Have A Nice Day Ahead. :::")

print ("About:")

with open('About.txt','r') as infile:

show = infile.read()

print (show)

with open(filename,'a+') as ap:

#rot13() function is called for encoding

enc = rot13(user\_name.lower())

re\_new = [acc\_no,enc,str(Pin),str(net\_balance),time.strftime('%d-%b-%Y at %I:%M %p'),address]

w = csv.writer(ap)

w.writerow(re\_new)

ap.close()

return

#Deposit funtion starts when called by atm function

def deposit(Net\_balance, address):

clear = ('cls' if os.name == 'nt' else 'clear')

global net\_balance

print(":: Deposit ::")

try:

deposit\_amount = input("Enter Amount In Rupees: ")

#Check for negetive values

if float(deposit\_amount) >= 0.0:

#check for extra large amount

#limits amount towards power of e

if (len(deposit\_amount) > 14) or ((len(str(float(deposit\_amount)+net\_balance))) > 14):

os.system(clear)

print (':: Amount Limit Exceeded! ::')

return

#Deposit amount is incremented in counter

else:

net\_balance += float(deposit\_amount)

os.system(clear)

MSG = "You Have Successfully Depositted An Amount Of Rs "+str(deposit\_amount)+"\n\nYour Net Account Balance is Rs "+str(net\_balance)

msg = sendmail(address, MSG)

os.system(clear)

if not (msg == True): print(msg)

print(":: You Have Successfully Depositted An Amount Of Rs",deposit\_amount,"::",'\n')

return

elif float(deposit\_amount) < 0.0:

os.system(clear)

#If user inputs negetive amount

print (":: Please Enter Right Amount! ::\n")

return deposit(net\_balance, address)

else:

os.system(clear)

print (":: Please Enter Right Amount! ::\n")

return deposit(net\_balance, address)

except ValueError:

os.system(clear)

print (":: Please Enter Right Amount! ::\n")

return deposit(net\_balance, address)

#deposit funtion starts when called by atm function

def withdraw(Net\_balance, address):

clear = ('cls' if os.name == 'nt' else 'clear')

global net\_balance

print(":: Withdraw ::")

#If amount is zero returns to atm function

if float(net\_balance) <= 0.0:

print (":: Withdrawl Impossible! ::\n:: Your Account Balance = Rs",net\_balance,"::","\n:: Please Deposit Amount First! ::\n")

return

else:

try:

with\_draw = input("Enter Amount In Rupees: ")

os.system(clear)

#If user inputs negetive amount

if float(with\_draw) < 0.0:

os.system(clear)

print (":: Please Enter Right Amount! ::\n")

return withdraw(net\_balance, address)

#Checks if amount in withdraw is less than amount in counter

elif float(with\_draw) <= net\_balance:

net\_balance -= float(with\_draw)

MSG = "You Have Successfully Withdrawn An Amount Of Rs "+str(with\_draw)+"\n\nYour Net Account Balance is Rs "+str(net\_balance)

msg = sendmail(address, MSG)

os.system(clear)

if not (msg == True): print(msg)

print(":: You Have Successfully Withdrawn An Amount Of Rs",with\_draw,"::",'\n')

return

else:

os.system(clear)

print (":: Withdrawl Impossible! ::\n:: Your Acount Balance = Rs",net\_balance,"::","\n")

return withdraw(net\_balance, address)

except ValueError:

os.system(clear)

print (":: Please Enter Right Amount! ::\n")

return withdraw(net\_balance, address)

def change\_pin(Pin, address):

clear = ('cls' if os.name == 'nt' else 'clear')

os.system(clear)

pin\_count = 0

print(":: Create Your Own Pin....::")

while pin\_count != 3:

print(":: Entries left :",(3-pin\_count),"::")

pin = str(gp ("Enter 4-Digit Pin : "))

os.system(clear)

if (len(pin) == 4) and (pin.isdigit() == True):

if not pin == Pin:

os.system(clear)

confirm\_pin = str(gp ("Confirm Pin : "))

if pin == confirm\_pin:

Pin = pin

os.system(clear)

MSG = "You Have Successfully Changed Your Pin"

msg = sendmail(address, MSG)

os.system(clear)

if not (msg == True): print(msg)

print(':: Pin Changed Successfully! ::\n')

return(Pin)

else:

os.system(clear)

print(":: Pin Change Unsuccessful! ::")

print (":: Your Pin Did Not Match! ::\n")

pin\_count +=1

else:

pin\_count += 1

os.system(clear)

print(":: Pin Change Unsuccessful! ::")

print(":: Please Enter A New Pin ::\n")

else:

pin\_count += 1

os.system(clear)

print(":: Pin Change Unsuccessful! ::")

print(":: Invalid Pin! ::\n")

return(Pin)

def amount\_transfer(account\_no, balance, acc\_no, address):

import time,datetime

clear = ('cls' if os.name == 'nt' else 'clear')

os.system(clear)

d = data()

filename = join()

amount = 0.0

print (":: Amount Transfer ::")

Inactive\_account = str('#'+account\_no)

if Inactive\_account in d.keys():

os.system(clear)

print (":: Provided Account Number Is Not Active! ::")

return amount

elif account\_no in d.keys():

try:

amount = input("Enter Amount In Rupees: ")

if float(amount) < 0.0 or ('-' in amount):

os.system(clear)

print (":: Please Enter Right Amount! ::\n")

return amount\_transfer(account\_no, balance, acc\_no, address)

elif float(amount) > float(balance):

os.system(clear)

print (":: Amount Can Not Be Transferred! ::\n:: Your Acount Balance = Rs",balance,"::","\n")

return amount\_transfer(account\_no, balance, acc\_no, address)

else:

os.system(clear)

print(":: Account Number :",account\_no,"::")

print(":: Name :",d[account\_no][0],"::")

print(":: Amount Transfer = Rs","{:,} ::".format(float(amount)),"\n")

confirm = input("Please Confirm \n1. Yes \n2. No \n")

if (confirm == '1') or (confirm.lower().startswith('y')):

with open(filename,'a+') as ap:

#rot13() function is called for encoding

enc = rot13(d[account\_no][0])

current\_balance = balance

balance = str(float(d[account\_no][2]) + float(amount))

Message = str("Amount of 'Rs "+str(amount)+"' was received on "+str(time.strftime('%d-%b-%Y at %I:%M %p'))+", through Account Number: "+str(acc\_no))

re\_new = [account\_no,enc,d[account\_no][1],balance,d[account\_no][3],d[account\_no][4],Message]

w = csv.writer(ap)

w.writerow(re\_new)

ap.close()

os.system(clear)

MSG\_from = "You Have Successfully Transferred An Amount Of Rs "+str(amount)+" To A/C #"+str(account\_no)+"\n\nYour Net Account Balance Is Rs "+str(float(current\_balance) - float(amount))

MSG\_to = "You Have Received An Amount Of Rs "+str(amount)+" From A/C #"+str(acc\_no)+"\n\nYour Net Account Balance Is Rs "+str(float(balance))

sendmail(address, MSG\_from)

msg2 = sendmail(d[account\_no][4], MSG\_to)

os.system(clear)

if not (msg2 == True): print(msg2)

print(":: Amount Transferred Successfully! ::")

return amount

else:

amount = 0

os.system(clear)

print(":: Amount Transfer Unsuccessful! ::")

return amount

except ValueError as err:

os.system(clear)

print("Error :",err)

print(":: Please Enter Right Amount! ::\n")

return amount\_transfer(account\_no, balance, acc\_no, address)

else:

os.system(clear)

print (":: No Match Found! ::")

return amount

* **Python File** : Data.py

from encrypt import rot13

import os

import csv

#relative path for file

def join():

directory = "Data"

name = "usersdata.csv"

filename = os.path.join(directory, name)

return filename

#when 1 is entered from main(login\_user)

def data():

filename = join()

d = {}

new = ['Account Number','Name','PIN','Amount','Time','Email Address']

try:

#file size shorter than 13 bit

with open(filename, "a") as ap:

if (os.path.getsize(filename)) <= 0:

wr = csv.writer(ap)

wr.writerow(new)

ap.close()

print ("Please create an account first!")

return

else:

with open(filename, "r") as rd:

r = csv.reader(rd)

for indiv\_user\_info in r:

if (indiv\_user\_info == ['Account Number','Name','PIN','Amount','Time','Email Address']) or (indiv\_user\_info == []):

continue

else:

try:

#rot13() function is called for decoding

indiv\_user\_info[1] = rot13(indiv\_user\_info[1])

indiv\_user\_info[3] = float(indiv\_user\_info[3])

d[indiv\_user\_info[0]] = indiv\_user\_info[1],indiv\_user\_info[2],indiv\_user\_info[3],indiv\_user\_info[4],indiv\_user\_info[5],indiv\_user\_info[6]

except IndexError:

d[indiv\_user\_info[0]] = indiv\_user\_info[1],indiv\_user\_info[2],indiv\_user\_info[3],indiv\_user\_info[4],indiv\_user\_info[5],"None"

return d

except (IOError or FileNotFoundError):

os.mkdir("Data")

data()

* **Python File** : encrypt.py

#for encoding of name

def rot13(s):

chars = "abcdefghijklmnopqrstuvwxyz"

trans = chars[13:]+chars[:13]

rot\_char = lambda c: trans[chars.find(c)] if chars.find(c)>-1 else c

return ''.join( rot\_char(c) for c in s )

* **Python File** : login.py

#IMPORTS

#########################################

## USING PYTHON BUILT-IN LIBRARIES

from \_\_future\_\_ import print\_function

import time,datetime

import os, sys, csv, re

import random as rd

from getpass import getpass as gp

##USING SOURCE FILES

from ATM import atm

from encrypt import rot13

from Data import data,join

from acc\_no\_gen import account\_no\_gen, code

from send\_mail import sendmail

###########################################

#END OF IMPORTS

# Using input() in python 2 or 3

try:

# set raw\_input as input in python2

input = raw\_input

except:

pass

#Clear the working terminal

clear = ('cls' if os.name == 'nt' else 'clear')

#main funtion which calls further funtions,execution starts from here

def login\_user():

clear = ('cls' if os.name == 'nt' else 'clear')

#data funtion is called to check or make changes in it

d = data()

user = input("Select One : \n1. Login \n2. Create New Account \n3. Activate Account \n4. De-Activate Account \n0. Exit \n")

os.system(clear)

if not str(user).isdigit():

print ("Invalid Selection!")

return login\_user()

#login function called for further execution

if int(user) == 1:

os.system(clear)

login(d)

#new\_account function called for further execution

elif int(user) == 2:

os.system(clear)

new\_account()

elif int(user) == 3:

os.system(clear)

activate\_account()

elif int(user) == 4:

os.system(clear)

de\_active\_account()

#exit the main funtion

elif int(user) == 0:

print ("Good Bye!")

print ("About:")

with open('About.txt','r') as infile:

show = infile.read()

print (show)

#in case any other number is entered except those listed above

#recursion(main function called again)

else:

print ("Invalid Selection! '",user,"'")

return login\_user()

return

def login(d):

clear = ('cls' if os.name == 'nt' else 'clear')

os.system(clear)

user\_name = input("Login\nEnter Full Name : ")

entry = 0

if (d == None):

os.system(clear)

print ("Please create an account first!")

return new\_account()

for a in user\_name:

if ((ord(a) >= 65) and (ord(a) <= 90)) or ((ord(a) >= 97) and (ord(a) <= 122)) or (ord(a) == 32):

continue

else:

os.system(clear)

print ("Invalid User!")

return login\_user()

acc\_no = None

for item in d.keys():

if user\_name.lower() in d[item]:

acc\_no = item

break

else:

acc\_no = None

if acc\_no == None:

os.system(clear)

print("Account not found/Invalid Name!")

return login\_user()

elif acc\_no.startswith('#'):

os.system(clear)

print("Account Is De-Activated!")

return login\_user()

#--Admin Block--

elif (user\_name.lower() == 'admin access'):

return admin\_block(acc\_no)

#users block

elif not (user\_name.lower() == 'admin access'):

user\_name\_l = user\_name.lower()

while int(entry) != 3:

print("Entries left :",(3-entry))

pin = str(gp("Enter 4-Digit Pin : "))

if pin == d[acc\_no][1]:

Pin = pin

Net\_balance = d[acc\_no][2]

History = d[acc\_no][3]

Mail\_address = d[acc\_no][4]

Message = d[acc\_no][5]

os.system(clear)

# Shows message at the top if there is any!

if (Message == "None"):

(None)

else:

print ("Message: ",Message)

return atm(user\_name,Net\_balance,Pin,History,acc\_no,Mail\_address)

else:

entry += 1

os.system(clear)

print ("Incorrect Pin!")

os.system(clear)

print ("Login Unsuccessful\n")

return login\_user()

else:

os.system(clear)

print ("Invalid User!")

return login\_user()

def new\_account():

clear = ('cls' if os.name == 'nt' else 'clear')

import time,datetime

filename = join()

user\_name1 = input("New Account\nEnter First Name : ")

os.system(clear)

user\_name2 = input("Enter Last Name : ")

if (user\_name1.isalpha() == False) or (user\_name2.isalpha() == False) or (user\_name1 == user\_name2):

os.system(clear)

print ("Invalid Name!")

return new\_account()

#auto-generated pin

auto\_gen\_pin = rd.randint(1000,9999)

os.system(clear)

full\_name = (user\_name1.lower())+' '+(user\_name2.lower())

acc\_no = account\_no\_gen(full\_name)

conf = code()

Mail\_address = input("Please Enter A Valid Email Address : ")

if not re.match(r"^[A-Za-z0-9\.\+\_-]+@[A-Za-z0-9\.\_-]+\.[a-zA-Z]\*$", Mail\_address):

os.system(clear)

confirm\_mail = "None"

print("\_\_\_\_INVALID-MAIL-ADDRESS\_\_\_\_")

else:

MSG = "Confirming mail address!"+"\n\n"+"Account Number : "+acc\_no+"\n\nYour Verification Code Is : '"+conf+"'"

confirm\_mail = sendmail(Mail\_address, MSG, "Confirmation Mail")

if (confirm\_mail == True):

os.system(clear)

print("Verification Code Has Been Sent To Your Email Address!")

conf\_code = input("Enter Provided Code : ")

if (conf\_code.lower() == conf.lower()):

confirm\_mail = Mail\_address

os.system(clear)

print("Email Address Verified!\n")

else:

os.system(clear)

print("Invalid Code!\nYour Email Could Not Be Verified.\nYou May Try Again Later!\n")

confirm\_mail = "None"

else:

os.system(clear)

print(confirm\_mail,'\n')

confirm\_mail = "None"

print("Your Auto-Generated Pin : ",auto\_gen\_pin)

confirm = input("Want To Use This Pin ? \n1. Yes \n2. No \n")

if (confirm == '1') or (confirm.lower().startswith('y')):

os.system(clear)

print ("Account Name :",user\_name1+' '+user\_name2,"\nAccount Number :",acc\_no,"\nPin :",auto\_gen\_pin)

confirm = input("Please Confirm \n1. Yes \n2. No \n")

if (confirm == '1') or (confirm.lower().startswith('y')):

os.system(clear)

with open(filename, "a+") as wr:

#rot13() function is called for encoding

enc = rot13(full\_name)

new = [acc\_no,enc,auto\_gen\_pin,'0.0',time.strftime('%d-%b-%Y at %I:%M %p'),confirm\_mail]

w = csv.writer(wr)

w.writerow(new)

wr.close()

MSG = "Dear "+str(full\_name.upper())+"!\n\tWelcome To YOB(YOUR OWN BANK) Service. Your account is successfully created. \n\tThanks for putting your trust on our service. \n\n\nFor any queries, feel free to contact our 24 hours costumer service at: yobfast.services@gmail.com"

vr = sendmail(Mail\_address, MSG)

os.system(clear)

if not (vr == True): print(vr)

print ("Account Created Successfully! \n")

return login\_user()

elif (confirm == '2') or (confirm.lower().startswith('n')):

os.system(clear)

print ("Account Not Created!")

return login\_user()

else:

os.system(clear)

print ("Account Not Created!")

return new\_account()

else:

os.system(clear)

pin\_count = 0

print("Create Your Own Pin....")

while pin\_count != 3:

print("Entries left :",(3-pin\_count))

pin = str(gp ("Enter 4-Digit Pin : "))

os.system(clear)

if (len(pin) == 4) and (pin.isdigit() == True):

os.system(clear)

confirm\_pin = str(gp ("Confirm Pin : "))

if pin == confirm\_pin:

os.system(clear)

print ("Account Name :",user\_name1+' '+user\_name2,"\nAccount Number :",acc\_no,"\nPin :",pin)

confirm = input("Please Confirm \n1. Yes \n2. No \n")

if (confirm == '1') or (confirm.lower().startswith('y')):

os.system(clear)

with open(filename, "a+") as wr:

#rot13() function is called for encoding

enc = rot13(full\_name)

new = [acc\_no,enc,pin,'0.0',time.strftime('%d-%b-%Y at %I:%M %p'),confirm\_mail]

w = csv.writer(wr)

w.writerow(new)

wr.close()

MSG = "Dear "+str(full\_name.upper())+"!\n\tWelcome To YOB(YOUR OWN BANK) Service. Your account is successfully created. \n\tThanks for putting your trust on our service. \n\n\nFor any queries, feel free to contact our 24 hours costumer service at: yobfast.services@gmail.com"

vr = sendmail(Mail\_address, MSG)

os.system(clear)

if not (vr == True): print(vr)

print ("Account Created Successfully! \n")

return login\_user()

elif (confirm == '2') or (confirm.lower().startswith('n')):

os.system(clear)

print ("Account Not Created!")

return login\_user()

else:

os.system(clear)

print ("Account Not Created!")

return new\_account()

else:

print ("Your Pin Did Not Match!")

pin\_count +=1

else:

pin\_count = pin\_count

os.system(clear)

print ("Invalid Pin!")

os.system(clear)

print ("Account Not Created!")

return login\_user()

def activate\_account():

clear = ('cls' if os.name == 'nt' else 'clear')

d = data()

filename = join()

user\_acc\_no = str(input('Enter 12-Digit Account Number : '))

os.system(clear)

if not user\_acc\_no.isdigit():

print('Invalid Account!')

return login\_user()

elif user\_acc\_no in d.keys():

print('Account Is Already Active!')

return login\_user()

elif user\_acc\_no.isdigit():

ch\_acc\_no = str('#'+user\_acc\_no)

if ch\_acc\_no in d.keys():

d[user\_acc\_no] = d.pop(ch\_acc\_no)

print ("Activate Account Name :",d[user\_acc\_no][0])

confirm = input("Please Confirm \n1. Yes \n2. No \n")

if (confirm == '1') or (confirm.lower().startswith('y')):

os.system(clear)

#over\_writing of existing file

with open(filename,"w") as rd:

r = csv.writer(rd)

r.writerow(['Account Number','Name','PIN','Amount','Time','Email Address'])

rd.close()

with open(filename,"a") as ow:

for item in d.keys():

items = rot13(d[item][0])

over\_write = [item,items,str(d[item][1]),str(d[item][2]),str(d[item][3]),str(d[item][4])]

o = csv.writer(ow)

o.writerow(over\_write)

ow.close()

print ("Account Activated Successfully! \n")

return login\_user()

elif (confirm == '2') or (confirm.lower().startswith('n')):

os.system(clear)

print ("Account Not Activated!")

return login\_user()

else:

os.system(clear)

print ("Account Not Activated!")

return login\_user()

else:

os.system(clear)

print('Account Does Not Exist')

return login\_user()

def de\_active\_account():

clear = ('cls' if os.name == 'nt' else 'clear')

d = data()

filename = join()

os.system(clear)

acc\_no = input("Account De-activate\nEnter Account Number : ")

if acc\_no in d.keys():

acc\_pin = str(gp("Enter 4-Digit Pin : "))

if acc\_pin == d[acc\_no][1]:

os.system(clear)

print ("De-activate Account :",d[acc\_no][0])

confirm = input("Please Confirm \n1. Yes \n2. No \n")

if (confirm == '1') or (confirm.lower().startswith('y')):

os.system(clear)

d[('#'+acc\_no)] = d.pop(acc\_no)

#over\_writing of existing file

with open(filename,"w") as rd:

r = csv.writer(rd)

r.writerow(['Account Number','Name','PIN','Amount','Time','Email Address'])

rd.close()

with open(filename,"a") as ow:

for item in d.keys():

items = rot13(d[item][0])

over\_write = [item,items,str(d[item][1]),str(d[item][2]),str(d[item][3]),str(d[item][4])]

o = csv.writer(ow)

o.writerow(over\_write)

ow.close()

print ("Account De-Activated Successfully! \n")

return login\_user()

elif (confirm == '2') or (confirm.lower().startswith('n')):

os.system(clear)

print ("Account Not De-Activated!")

return login\_user()

else:

os.system(clear)

print ("Account Not De-Activated!")

return login\_user()

else:

os.system(clear)

print ("Pin Did Not Match!")

return login\_user()

elif ("#"+acc\_no) in d.keys():

os.system(clear)

print("Account Is Already De-Active!")

return login\_user()

else:

os.system(clear)

print ("No match found!")

return login\_user()

def admin\_block(acc\_no):

clear = ('cls' if os.name == 'nt' else 'clear')

d = data()

pin = str(gp("Enter 4-Digit Pin : "))

if pin == d[acc\_no][1]:

del d[acc\_no]

os.system(clear)

print (time.strftime('Date:%d-%b-%Y \nTime:%I:%M %p Today:%A\n'))

print ("::: Welcome to YOB Admin Block! :::\n\n:: Select Option Provided Below ::")

ad = input("1. Number Of Users \n2. Active User Names \n3. Active Users Info. \n4. Users Acivity \n5. Find Account \n6. De-Activate Account\n0. Exit\n")

while ad != '0':

if ad == '1':

os.system(clear)

c\_user, i\_user = 0, 0

for users in d.keys():

if not users.startswith('#'):

c\_user += 1

else:

i\_user += 1

print(":: Users ::")

print("Active Users :",c\_user)

print("Inactive Users :",i\_user,'\n')

elif ad == '2':

os.system(clear)

c\_user = 0

print (":: Active User Names ::")

for users in d.keys():

if not users.startswith('#'):

c\_user += 1

print ("Active User",c\_user,':',d[users][0])

print('\n')

elif ad == '3':

os.system(clear)

print (":: Users Info ::")

for user\_info in d.keys():

if not user\_info.startswith('#'):

print ("Name =",d[user\_info][0],", Pin :",d[user\_info][1],", Amount :","{:,}".format(d[user\_info][2]))

print('\n')

elif ad == '4':

os.system(clear)

print (":: Users Acivity ::")

for user\_info in d.keys():

if not user\_info.startswith('#'):

print ("Account Number :",user\_info,"of Name :",d[user\_info][0],"was previously logged in on",d[user\_info][3])

print('\n')

elif ad == '5':

os.system(clear)

acc\_no\_find = str(input('Enter 12-Digit Account Number : '))

if acc\_no\_find in d.keys():

print("Account Status : Active")

print("Name :",d[acc\_no\_find][0])

print("Pin :",d[acc\_no\_find][1])

print("Amount :","{:,}\n".format(d[acc\_no\_find][2]))

elif ("#"+acc\_no\_find) in d.keys():

print("Account Status : Inactive")

print("Name :",d[("#"+acc\_no\_find)][0])

print("Pin :",d[("#"+acc\_no\_find)][1])

print("Amount :","{:,}\n".format(d[("#"+acc\_no\_find)][2]))

else:

os.system(clear)

print("Account Not Found!")

elif ad == '6':

os.system(clear)

return de\_active\_account()

ad = input("1. Number Of Users \n2. Active User Names \n3. Active Users Info. \n4. Users Acivity \n5. Find Account \n6. De-Activate Account\n0. Exit\n")

os.system(clear)

return login\_user()

else:

os.system(clear)

return login\_user()

try:

os.system(clear)

login\_user()

except Exception as exc:

os.system(clear)

print ("Some errors were encountered: %s" %exc)

print ("Sorry for inconvenience.\nGood bye!")

* **Python File** : send\_mail.py

from \_\_future\_\_ import print\_function

import os, sys

import smtplib

def sendmail(address, msg, sbj = "YOB BANK"):

clear = ('cls' if os.name == 'nt' else 'clear')

try:

try:

server = smtplib.SMTP('smtp.gmail.com', 587)

print ("Please wait...")

server.starttls()

server.login("yobfast.services@gmail.com", "pakistan100")

os.system(clear)

print ("Please wait....")

message = 'Subject: {}\n\n{}'.format(sbj, msg)

server.sendmail("yobfast.services@gmail.com", address, message)

os.system(clear)

print ("Please wait.....")

server.sendmail("yobfast.services@gmail.com", "admin.yob@gmail.com", str(address)+"\n"+str(msg))

os.system(clear)

print ("Please wait......")

server.quit()

return True

except Exception:

server = smtplib.SMTP('smtp.gmail.com', 587)

os.system(clear)

print ("Please wait.....")

server.starttls()

server.login("yobfast.services@gmail.com", "password100")

os.system(clear)

print ("Please wait......")

server.sendmail("yobfast.services@gmail.com", "admin.yob@gmail.com", str(address)+"\n"+str(msg))

os.system(clear)

print ("Please wait.......")

server.quit()

return "\_\_\_\_INVALID-MAIL-ADDRESS\_\_\_\_"

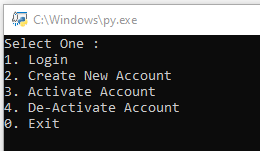
except Exception:

return "\_\_\_\_CONNECTION-TIMEDOUT\_\_\_\_\_"

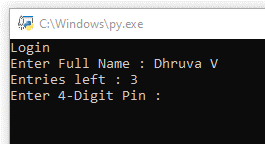
**AREA OF FOCUS / OPTIMIZATIONS**

* Check the number of users.
* Check the active accounts name.
* Check the accounts information.
* Check the accounts activity.
* Search accounts
* De-Activate account.

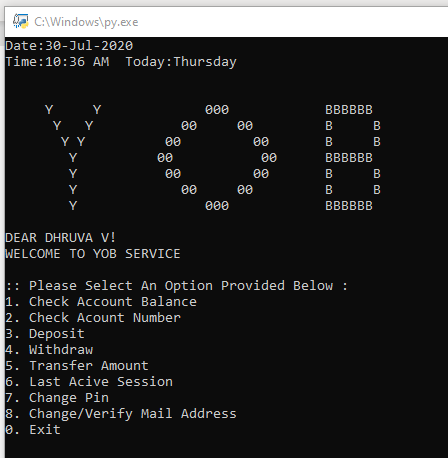
**RESULTS / CONCLUSION**



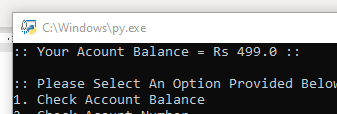
* **Login Menu**
* **Login Enter Username & Pin**



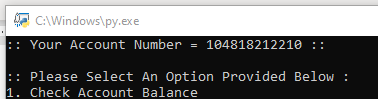
* **ATM Home Page**



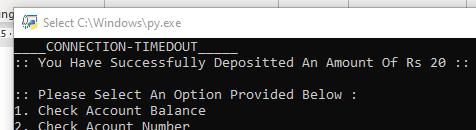
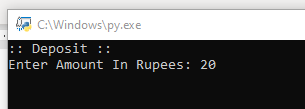
* **Account Balance**



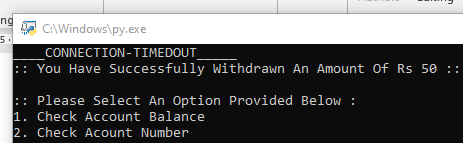
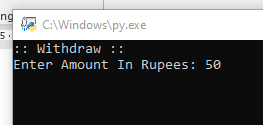
* **Check Account Number**



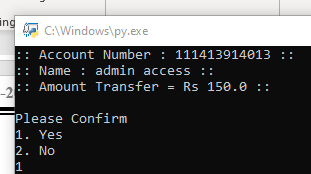
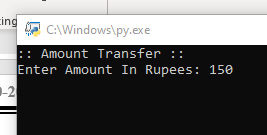
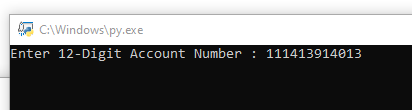
* **Deposit**



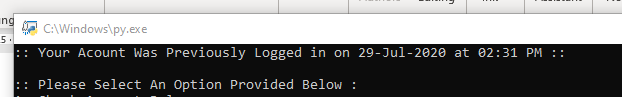
* **Withdraw**



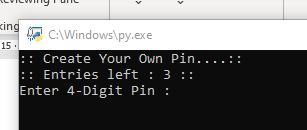
* **Transfer**



* **Last Active Session**



* **Change Pin**



Since basic functions of ATM using python was the main objective of the project, we quantified the results of the project in terms of its functionalities implementation. The specifications of the computer used are as follows:

* Intel Core i3 or greater processor
* 4GB of RAM
* Monitor
* Python 2 or Python 3 version
* Visual Studio Code latest version
* Keyboard
* Mouse

**REFERENCES**

The source code for the project can be found at : <https://github.com/Dhruvawara/Python-ATM-project>

* <https://docs.python.org/3/>
* <https://www.elprocus.com/automatic-teller-machine-types-working-advantages/>