

GROUPE SOCIÉTÉ GÉNÉRALE
SG RECRUITMENT PROCESS

SIMPLE BANKING SYSTEM

A Project Report

Submitted by

Shakthivel MURUGAVEL

of

E5 - ARTIFICIAL INTELLIGENCE AND CYBERSECURITY

Under the Guidance of

Madame Therese DE LA BARTHE



93160, Noisy-le-Grand, France

21st December 2023

TABLE OF CONTENTS

TITLE	PAGE NO.
1. REQUIREMENT ANALYSIS	03
2. DESIGN AND ARCHITECTURE	04
3. DEVELOPMENT	04
3.1 MVP FEATURES	06
3.2 ADDITIONAL FEATURES	06
3.3 UX IMPROVEMENT	06
3.4 CYBERSECURITY	06
3.5 CODE CLEAN UP	06
4. TESTING	06
4.1 UNIT TESTING	06
5. DEPLOYMENT	07
6. CONCLUSION	07
7. APPENDIX	08

PART 1

REQUIREMENT ANALYSIS

To create a personal bank account experience. Go for the simplest solution.

Requirements

- Deposit and Withdrawal
- Account statement (date, amount, balance)
- Statement printing

US1: Deposit

In order to save money as a bank client I want to make a deposit in my account

US 2: Withdraw

In order to retrieve some or all of my savings as a bank client I want to make a withdrawal from my account

US 3: Statement

In order to check my operations as a bank client, I want to see the history (operation, date, amount, balance) of my operations.

“Le plus simple est le mieux.”

PART 2

DESIGN AND ARCHITECTURE

The requirement is straight forward. As a simple solution is expected, a simple approach with python is used in this project.

The user data storage is done in CSV file. Different thoughts were put into the structure of the database. Finally, appending every transaction of the user cell wise horizontally and new users vertically was implemented.

PART 3

DEVELOPMENT

3.1 MVP FEATURES

All the basic features of user creation, user login, deposit, withdraw and statement printing was successfully implemented.

3.2 ADDITIONAL FEATURES

Two new features of downloading the user transactions as a file and displaying the current account balance was added to the user experience.

3.3 UX IMPROVEMENT

The user experience is improved compared to the first version (found in git commits). There are no abrupt exits in the script. Provides the user with a smooth experience. A few newline commands are also added in the code for better readability and User Interface (UI).

3.4 CYBERSECURITY

Buffer exploit

It is a common cyber-attack done by overloading the user input. A buffer overflow may lead to access to other memory spaces. This program is protected from it by adding a limit to user inputs wherever necessary.

Hidden Password

Password is hidden in the terminal to protect it from prying eyes.

Encryption

Password is encrypted and saved in the customer database to avoid password leaks.

3.5 CODE STANDARDS

- Unwanted code is removed.
- Keep it Short and Simple: Code is made short wherever possible.
- Don't Repeat Yourself (DRY) principle is used. Repetitive code is put in functions and re-called.
- Naming conventions is followed for better code representation and readability.
- Try catch is used wherever necessary.

PART 4

TESTING

UNIT TESTING SCENARIOS

All usual cases are well tested. Works perfectly. The edge cases are below:

1. Login without creating account first.

```
Are you an existing customer ? (y/n) : y
Welcome back, Customer. Please login.
Enter your email : a
Enter your password :
Email not found. Please create an account or contact customer support.
```

2. Other inputs other than y/n

```
Email not found. Please create an account or contact customer support.

Are you an existing customer ? (y/n) : f
Invalid input
Are you an existing customer ? (y/n) : 1
Invalid input
Are you an existing customer ? (y/n) : &
Invalid input
Are you an existing customer ? (y/n) : n

Hello! new customer
Let's create an account for you.
```

3. Password mismatch

```
Hello! new customer
Let's create an account for you.

Enter your Surname (MAX 50 Chars) : a
Enter your name (MAX 50 Chars) : b
Enter your email (MAX 50 Chars) : c
Create a password (MAX 50 Chars) :
Confirm your password :
Password mismatch. Try again.

Let's create an account for you.

Enter your Surname (MAX 50 Chars) : 
```

4. Invalid input

```
What would you like to do today ?  
1. Deposit  
2. Withdraw  
3. Print Statement  
4. Download Statement  
5. Show Balance  
6. Logout  
  
p  
  
Invalid input
```

```
Enter the amount you would like to deposit : pp  
  
Invalid input
```

```
Enter the amount you would like to withdraw: 2345.1234  
  
Insufficient balance
```

PART 5 DEPLOYMENT

Link to Github:

https://github.com/Shakthi1109/SG_Shakthi_Banque

Link to Demo:

<https://youtu.be/94L9PUaEKak>

PART 6 CONCLUSION

All the desired functionalities and some additional functionalities are added. A Cybersecurity component is infused with the development to improve and secure the program. The program is tested in most scenarios to avoid breaks or misinterpretation. Unit test cases are tested and passed. The simple banking program works successfully.

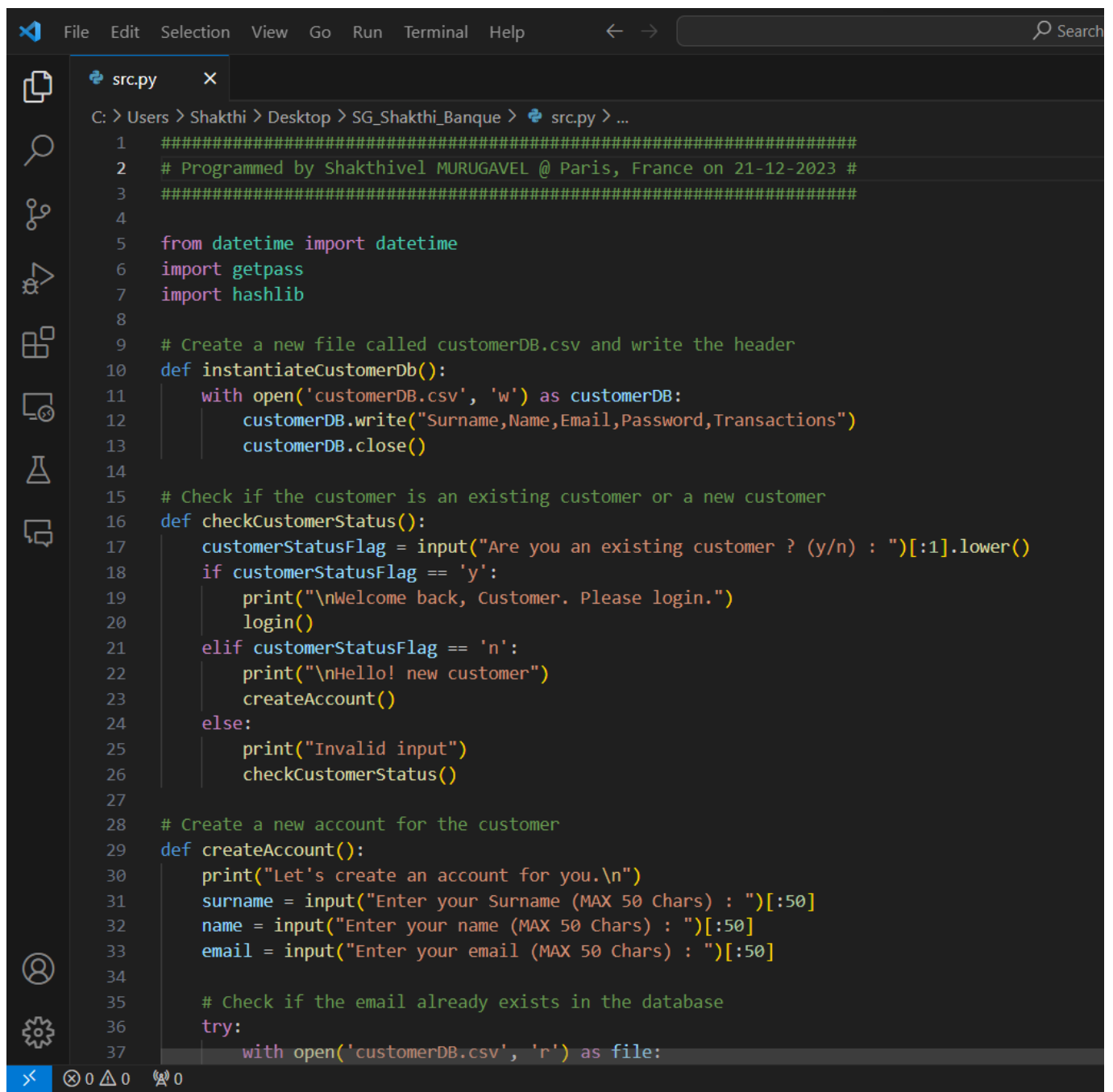
PART 7

APPENDIX

```
PS C:\Users\Shakthi\Desktop\SG_Shakthi_Banque> python src.py

#####
Welcome to Shakthi Bank
#####

Are you an existing customer ? (y/n) : 
```



```
src.py
C: > Users > Shakthi > Desktop > SG_Shakthi_Banque > src.py > ...
1 #####
2 # Programmed by Shakthivel MURUGAVEL @ Paris, France on 21-12-2023 #
3 #####
4
5 from datetime import datetime
6 import getpass
7 import hashlib
8
9 # Create a new file called customerDB.csv and write the header
10 def instantiateCustomerDb():
11     with open('customerDB.csv', 'w') as customerDB:
12         customerDB.write("Surname,Name,Email>Password,Transactions")
13         customerDB.close()
14
15 # Check if the customer is an existing customer or a new customer
16 def checkCustomerStatus():
17     customerStatusFlag = input("Are you an existing customer ? (y/n) : ")[:1].lower()
18     if customerStatusFlag == 'y':
19         print("\nWelcome back, Customer. Please login.")
20         login()
21     elif customerStatusFlag == 'n':
22         print("\nHello! new customer")
23         createAccount()
24     else:
25         print("Invalid input")
26         checkCustomerStatus()
27
28 # Create a new account for the customer
29 def createAccount():
30     print("Let's create an account for you.\n")
31     surname = input("Enter your Surname (MAX 50 Chars) : ")[:50]
32     name = input("Enter your name (MAX 50 Chars) : ")[:50]
33     email = input("Enter your email (MAX 50 Chars) : ")[:50]
34
35     # Check if the email already exists in the database
36     try:
37         with open('customerDB.csv', 'r') as file:
```


Are you an existing customer ? (y/n) : y

Welcome back, Customer. Please login.

Enter your email : shak@xyz.com

Enter your password :

*****Login successful*****

Welcome MURU shak

What would you like to do today ?

1. Deposit
2. Withdraw
3. Print Statement
4. Download Statement
5. Show Balance
6. Logout

1

Enter the amount you would like to deposit : 100.9

*****Deposit successful*****

Would you like to do another transaction ? (y/n) : y

Surname								
	A	B	C	D	E	F	G	H
1	Surname	Name	Email	Password	Transactions			
2	a	b	c	18ac3e7343f016890c51	Deposit - 21/12/2023 - 0 - 0	Deposit - 21/12/2023 - 50.0 - 50.0	Withdraw - 21/12/2023 - 3.0 - 47.0	
3	p	q	r	043a718774c572bd8a25	Deposit - 21/12/2023 - 0 - 0	Deposit - 21/12/2023 - 100.0 - 100.0	Deposit - 21/12/2023 - 0.0 - 100.0	Withdraw - 21/12/2023 - 4.0 - 96.0
4	x	y	z	594e519ae499312b2943	Deposit - 21/12/2023 - 0 - 0	Deposit - 21/12/2023 - 90.0 - 90.0	Withdraw - 21/12/2023 - 5.0 - 85.0	
5								

```
Name: m
Surname: m
Email: m
Transactions:
Operation - Date - Amount - Balance
Deposit - 21/12/2023 - 0 - 0
Deposit - 21/12/2023 - 100.0 - 100.0
Deposit - 21/12/2023 - 1.0 - 101.0
Withdraw - 21/12/2023 - 12.1 - 88.9
Withdraw - 21/12/2023 - 33.3 - 55.6

*****End of Statement*****
```

****Thank you****