SWAT 2

Class: Ecc108, Ecc201

Banking System Project Report

By: Abdullah, Amjad, Mahmoud, Omar, Qassim

Introduction:

The banking system project is like any other banking system around the world. Our project was to create a functional banking system that could act as a perfect replica of any other banking system. Our goal for this project was to be able to use pythonic language to code a system where the user could complete various functions such as creating an account, logging into the account, withdrawing money, depositing money, transferring money from one account to the other and lastly to check current balance.

Procedure/Planning:

A diagram of a flowchart

Description automatically generated

Our first goal when tackling this project was to create a UML and flowchart in order to better understand the question given to us. Thus, we began by creating a UML and flowchart which are shown below.

A white board with writing on it

Description automatically generated

After completing this step, we began to assign roles for the code keeping in mind which member was best suited for their role. We then decided which modules and tools we would use for the biggest part of the code, which was which filing system/storage system we would use. We were deciding between JSON module or CSV and eventually due to more members being familiar with CSV files we decided to use CSV files to store the information of the banking details

Conflicts:

In terms of rising problems during the creation of this program, we did not come across as many as was expected. Due to the familiarity of the program with a previous program that each of us had created we did not face any major issues. However, it must be stated that the assigning role’s part did not go as smoothly as we would have liked. Each of us had our own goals in mind, however this was causing issues with the harmony of the team and the overall betterment of the program. As for other problems we faced were more programming related, errors for syntaxes, readability, and an overall dynamic code. Due to our different styles of coding, there were some clashes with the writing of the code. For the most obvious conflict almost any programmer would face are the functions and how to use proper syntaxes/functions.

Code Explanation:

A screen shot of a computer

Description automatically generated

First of all we created a csv file named ‘demo.csv’ after that we defined the order and names of the columns that will be used in the csv file by naming it fieldnames it has number, name, password , balance.

A screen shot of a computer code

Description automatically generated

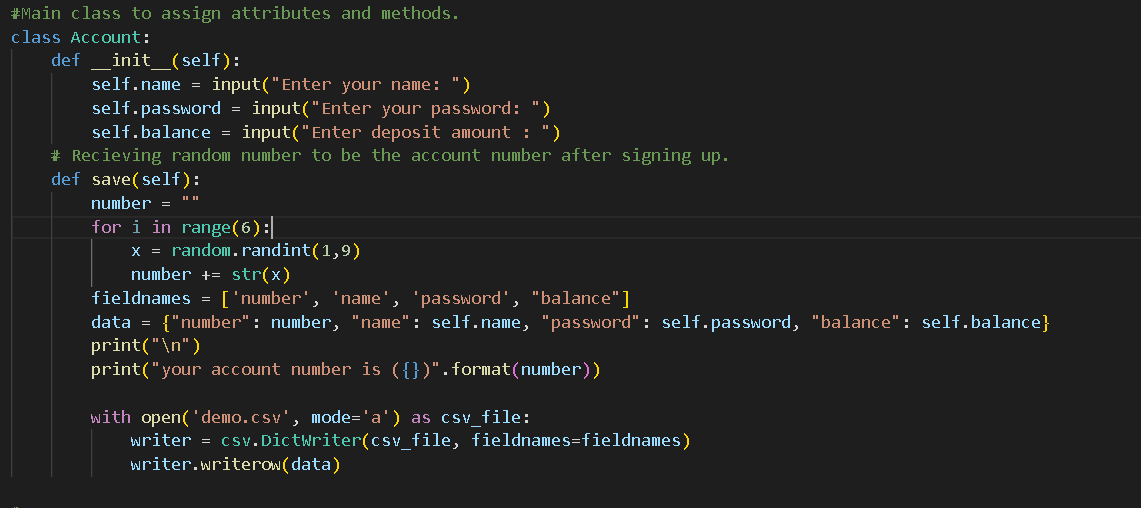
- This block opens the same csv file, but in read mode (“r”)

- A ‘csv DictReader’ is created which reads the rows of the csv file into dictionaries where the keys are column names

- The ‘rows = list (reader)’ line converts the reader object into all list of dictionaries

- An ‘index is a variable is initialized to 0

On the last part in photo this code defines if the current headers do not match the expected headers, the correct headers are appended to the file witch means that if the headers (column names) currently present in the csv file do not match the expected headers which are (numbers, name , password, balance ) then the correct headers will be added to the file. This ensures that the csv file consistently has the expected structure with the column names.



As you can see here that the first thing, we start by defining account as a class method after that the ‘\_\_innit\_\_’ method is called when a new instance of the account class is created it prompts the user to input their names, password, and initial balance.

After that the saving method this method generates a random 6 digits number to serv the account number it creats a dictionary data containing the account information which has (number, name, password, balance) then it will print the account number to the console then it will open the file ‘demo csv’ in a append mode and writes the information to it using a csv.DicWriter.

A screen shot of a computer program

Description automatically generated

On this part of the code, we defined a class named bank that has name then we update a specific cell in a csv file, it reads the csv file in the specified rows and writes the update dtata back to the csv file.

On the deposit method this part handles depositing money into the account it reads the csv file the update the balance by adding the deposited amount that we want

Withdraw method handles withdrawing money form the account it reads the csv filr finds the row corresponding to the provided account number and checks if the balance is right for withdrawal and updtaes the balance by substracting the withdrawn amount and then it will call the update\_csv method is to update the new amount on the file.

A computer screen shot of a code

Description automatically generated

This method part of the bank represents a transfer of the money from one account to another, ‘account no’ which is the account number of the sender, ‘receiver’ is the account number of the transfer, and ‘amount’ which is the amount to transfer

With open (“demo.csv”, “r”) this block opens the csv file in read mode then it reads the csv content using ‘csv.Dictreader ‘and stores the rows in a list and an index is a variable is to keep track of the current row in index

After that for method it loops throught each rows of the csv file then it checks if the account number in the current row matches the sender account number, if the balance in the sender account is enough for the transfer it calculates the new balance then it prints and update the new balance on the csv file

On the last line of the code explains after processing the sender account, it checks if the account number in the current row matches the receiver account number if it does it calculate the new balance after adding the transferred amount and finally updates It to the csv file.

A screen shot of a computer program

Description automatically generated

Finally, the sign up and log in page first of all we created a bank class and named it SWATT BANK

After that we do the sign up and log in pages

Inside the ‘main’ function on the second part on the photo it defines as a loop started to present a signup, login and exit options to the user.

-the signup comes if the user chooses to sign up (enters “1”) an instance of the account class is created the user to input their details and then it saves

-Log in

If the user chooses to login (enters “2”), the user requested to enter their account number and password.

After successful log in, the program displays the user balance and presents opstions for the banking operations which Is deposit, withdrawal, transfer or logout

When the user enters (enters “3”) it will transfer the the money to any account you want

Finally, the logout and exit

If the user chooses to logout the program breaks out of the loop abd returns to the main menu and if the user chooses to exit (enter “3”) the program exits the loop and ends, then u print it so it can appear on the terminal.

Conclusion:

With everything being said and done, despite all the issues, the solutions, the difficulties, we were able to produce a satisfactory code that takes care of all the goals we had in mind and were mentioned above. However, if we could change one thing, we were hoping to be able to create a GUI for the program as that was something that would elevate the program to another level, however due to time constraints this was not possible. If we were to do this program again, we would try to create a GUI to make the program look better but not only that, but it would allow the program to be more user friendly. Despite the lack of the GUI the program is still fully functional and does everything that it needs to do, and we were assigned. All in all, the program was successfully completed and is functioning therefore as a group we have completed the task at a satisfactory level.