Final Project 2k24

Team Members:

- Talha Aamir
- Usman Furqan
- Sahir Ali

Banking Project in Python

Problem:

Customer experience is an integral part of a bank's operations. That's why banks focus a lot on improving customer experience by removing hassles and enhancing the facilities they provide. Opening a new account in a bank usually requires a person to visit the bank, fill out a form, and submit the necessary papers. All of these tasks take up a lot of time and dampen the overall customer experience. Moreover, many people have to take time out of their schedules to go to a bank.

Solution :

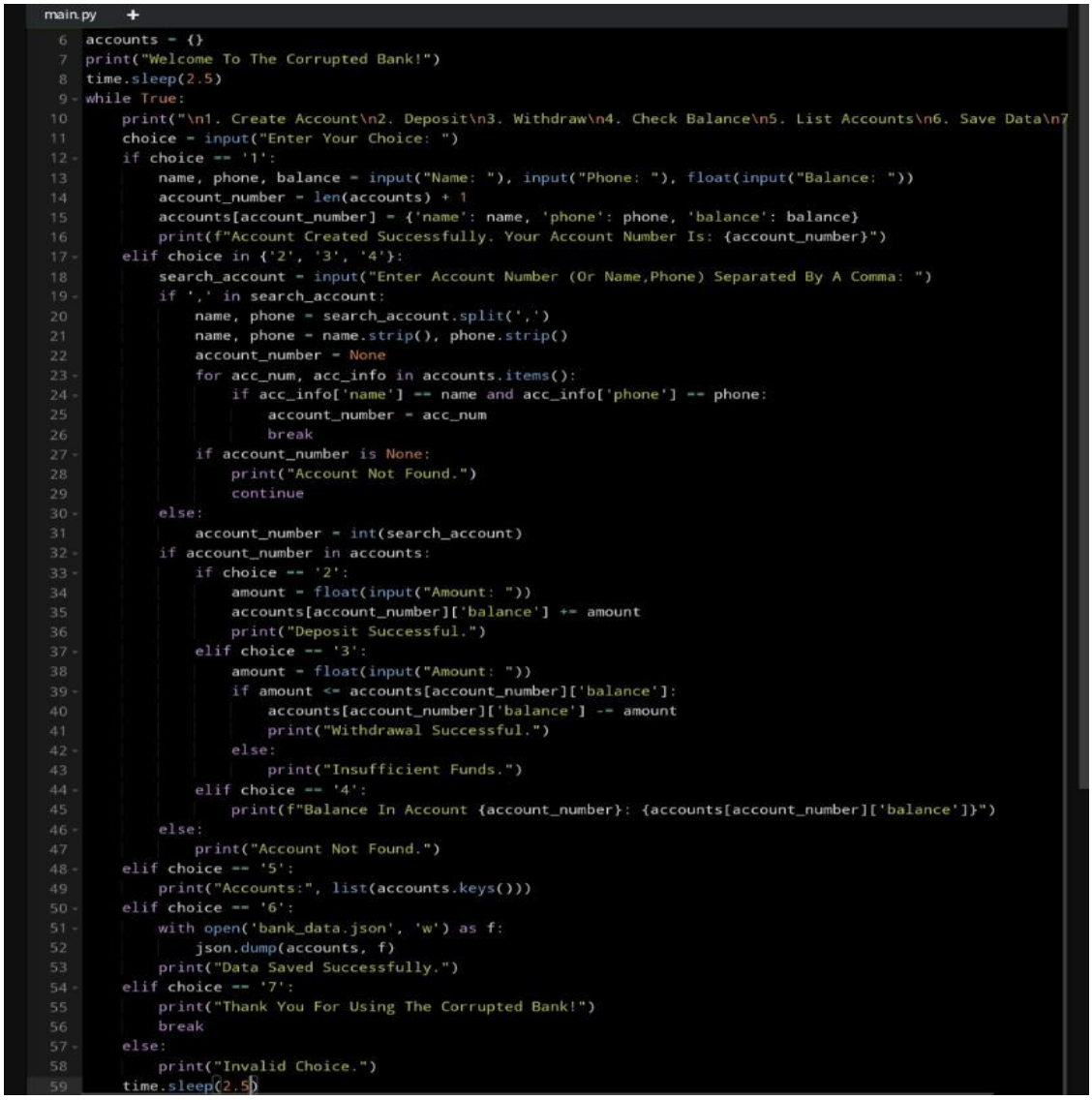
We have solve this problem by creating a software solution where people can sign up and open a new account in a bank digitally. This way, the person wouldn't have to visit the bank physically and thus, would save a lot of time and effort. The banking management system can also allow the user to make transactions, deposit and withdraw funds, and check the account balance. Most importantly our software can be use in both phase from customer perspective and also from employee perspective.

Functions Used

- **Input(): Used to take input from the user via the command line.**
- **print(): Used to display output to the user in the command line.**
- **len(): Used to get the length of an object, such as a list or a dictionary.**
- **split(): Used to split a string into a list of substrings based on a delimiter.**
- **time.sleep(): Used to pause the execution of the program for a specified number of seconds.**
- **float(): Used to convert a string or a number to a floating-point number.**
- **int(): Used to convert a string or a number to an integer.**
- **json.dump(): Used to write JSON data to a file.**

- **open(): Used to open a file.**
- **.append(): Used to add an item to the end of a list.**
- **.items(): Used to iterate over the items of a dictionary.**
- **.get(): Used to get the value associated with a specified key in a dictionary, with an optional default value if the key is not found.**
- **break: Used to exit from a loop prematurely.**
- **list(): Used to convert an iterable to a list.**

Screenshot of Code:



```
main.py +
accounts = {}
print("Welcome To The Corrupted Bank!")
time.sleep(2.5)
while True:
    print("\n1. Create Account\n2. Deposit\n3. Withdraw\n4. Check Balance\n5. List Accounts\n6. Save Data\n7
    choice = input("Enter Your Choice: ")
    if choice == '1':
        name, phone, balance = input("Name: "), input("Phone: "), float(input("Balance: "))
        account_number = len(accounts) + 1
        accounts[account_number] = {'name': name, 'phone': phone, 'balance': balance}
        print(f"Account Created Successfully. Your Account Number Is: {account_number}")
    elif choice in {'2', '3', '4'}:
        search_account = input("Enter Account Number (Or Name,Phone) Separated By A Comma: ")
        if ',' in search_account:
            name, phone = search_account.split(',')
            name, phone = name.strip(), phone.strip()
            account_number = None
            for acc_num, acc_info in accounts.items():
                if acc_info['name'] == name and acc_info['phone'] == phone:
                    account_number = acc_num
                    break
            if account_number is None:
                print("Account Not Found.")
                continue
        else:
            account_number = int(search_account)
        if account_number in accounts:
            if choice == '2':
                amount = float(input("Amount: "))
                accounts[account_number]['balance'] += amount
                print("Deposit Successful.")
            elif choice == '3':
                amount = float(input("Amount: "))
                if amount <= accounts[account_number]['balance']:
                    accounts[account_number]['balance'] -= amount
                    print("Withdrawal Successful.")
                else:
                    print("Insufficient Funds.")
            elif choice == '4':
                print(f"Balance In Account {account_number}: {accounts[account_number]['balance']}")
        else:
            print("Account Not Found.")
    elif choice == '5':
        print("Accounts:", list(accounts.keys()))
    elif choice == '6':
        with open('bank_data.json', 'w') as f:
            json.dump(accounts, f)
        print("Data Saved Successfully.")
    elif choice == '7':
        print("Thank You For Using The Corrupted Bank!")
        break
    else:
        print("Invalid Choice.")
    time.sleep(2.5)
```

Code Explained:

Let's break down the code further by explaining each function, variable, and other elements used:

1. **Variables:**
- `accounts`: This dictionary stores information about bank accounts. Each account is identified by a unique account number, and the corresponding value is a dictionary containing the account holder's name, phone number, balance, and transaction history.
- `choice`: This variable stores the user's choice from the menu.

2. **Functions:**
- None. The code primarily consists of procedural logic without defining any separate functions.

3. **Modules:**
- `json`: This module provides functions for encoding and decoding JSON data. It's used to save the `accounts` dictionary to a JSON file.
- `time`: This module provides various time-related functions. Here, it's used to add a delay between interactions to improve readability.

4. **Main Program Loop:**
- The `while True` loop ensures that the program continues running until the user chooses to exit (`choice == '7'`).

5. **Menu Display:**
- The `print` function displays a menu with options for the user to choose from.

6. **User Input:**
- The `input` function is used to prompt the user to enter their choice and other required information (e.g., name, phone number, balance, account number, amount).

7. **Conditional Statements (if-elif-else):**
- These statements check the user's choice and execute different parts of the code based on the selected option.

8. **Dictionary Manipulation:**
- The `accounts` dictionary is modified throughout the code to create accounts, deposit or withdraw money, and update account balances and transaction history.

9. **File I/O:**

- The `with open('bank_data.json', 'w') as f` block writes the contents of the `accounts` dictionary to a JSON file named "bank_data.json" when the user chooses to save data (`choice == '6'`).

10. **Delay**:
- The `time.sleep(1)` function adds a one-second delay after each interaction to improve readability.

That covers all the essential elements used in the code. It's a simple program that manages bank accounts through a basic command-line interface. Users can perform various operations such as creating accounts, depositing and withdrawing money, checking balances, listing accounts, saving data, and exiting the program.

Content:

1. **Menu Display**:

   - The program starts by showing a list of options for what you can do.

   - It’s like a menu in a restaurant where you choose what you want to eat.

2. **Account Creation (Option 1)**:

   - If you choose to create an account:

   - You’ll be asked for your name, phone number, and how much money you want to start with.

   - This information is saved as your account.

3. **Deposit Money (Option 2)**:

   - If you choose to deposit money:

   - You need to tell the program your account number.

   - Then, you say how much money you want to add to your account.

- **The program adds that money to your account balance.**

4. ****Withdraw Money (Option 3)**:**
   - **If you want to take out money:**
   - **You need to provide your account number.**
   - **Then, you say how much money you want to take out.**
   - **The program checks if you have enough money in your account.**
   - **If you do, it takes out the money you asked for.**

5. ****Check Balance (Option 4)**:**
   - **If you want to check how much money you have:**
   - **You provide your account number.**
   - **The program tells you how much money is in your account.**

6. ****List Accounts (Option 5)**:**
   - **If you want to see a list of all the account numbers:**
   - **The program shows you all the account numbers.**

7. ****Save Data (Option 6)**:**

- **If you want to save all the account information:**
- **The program saves everything to a file so you can come back later and see it.**

8. ****Exit (Option 7)**:**
   - **If you're done using the program:**
   - **You can choose to exit, and the program stops.**

That's pretty much it! The program helps you manage your bank account by letting you add money, take out money, and check your balance, just like a real bank.

This program is designed for both customers and bank staff. It provides basic functionalities for managing bank accounts, such as creating accounts, depositing and withdrawing money, checking balances, listing accounts, and saving data.

Customers can use it to perform transactions on their accounts, while bank staff can use it to assist customers with their banking needs, such as checking account details or performing transactions on behalf of customers.

Overall, it's a simple banking system that can be used by both customers and bank staff alike.