PYTHON BANK ACCOUNT MANGEMENT SYSTEM

INTRODUCTION

A Banking Management System is a software application designed to manage basic banking operations in a digital manner.

This Python-based project allows users to create bank accounts and perform operations such as deposit, withdrawal, money transfer, and viewing transaction history.

The system uses file handling to store data permanently, ensuring that user information and transaction records are preserved even after the program is closed.

PROBLEM STATEMENT

Manual banking processes are time-consuming and prone to human errors. This creates a need for a simple, secure, and automated banking system that can efficiently handle basic operations such as account management and transactions.

OBJECTIVE

- To design and develop a simple banking management system using Python
- To automate basic banking operations such as account creation, deposit, withdrawal, and money transfer
- To implement secure user authentication using username and password
- To store account and transaction data using file handling for data persistence
- To provide an easy-to-use, menu-driven interface for users

SYSTEM EXECUTION STEPS

Step 1

User Registration

Step 2

Secure Login

Step 3

Account Operations

Step 4

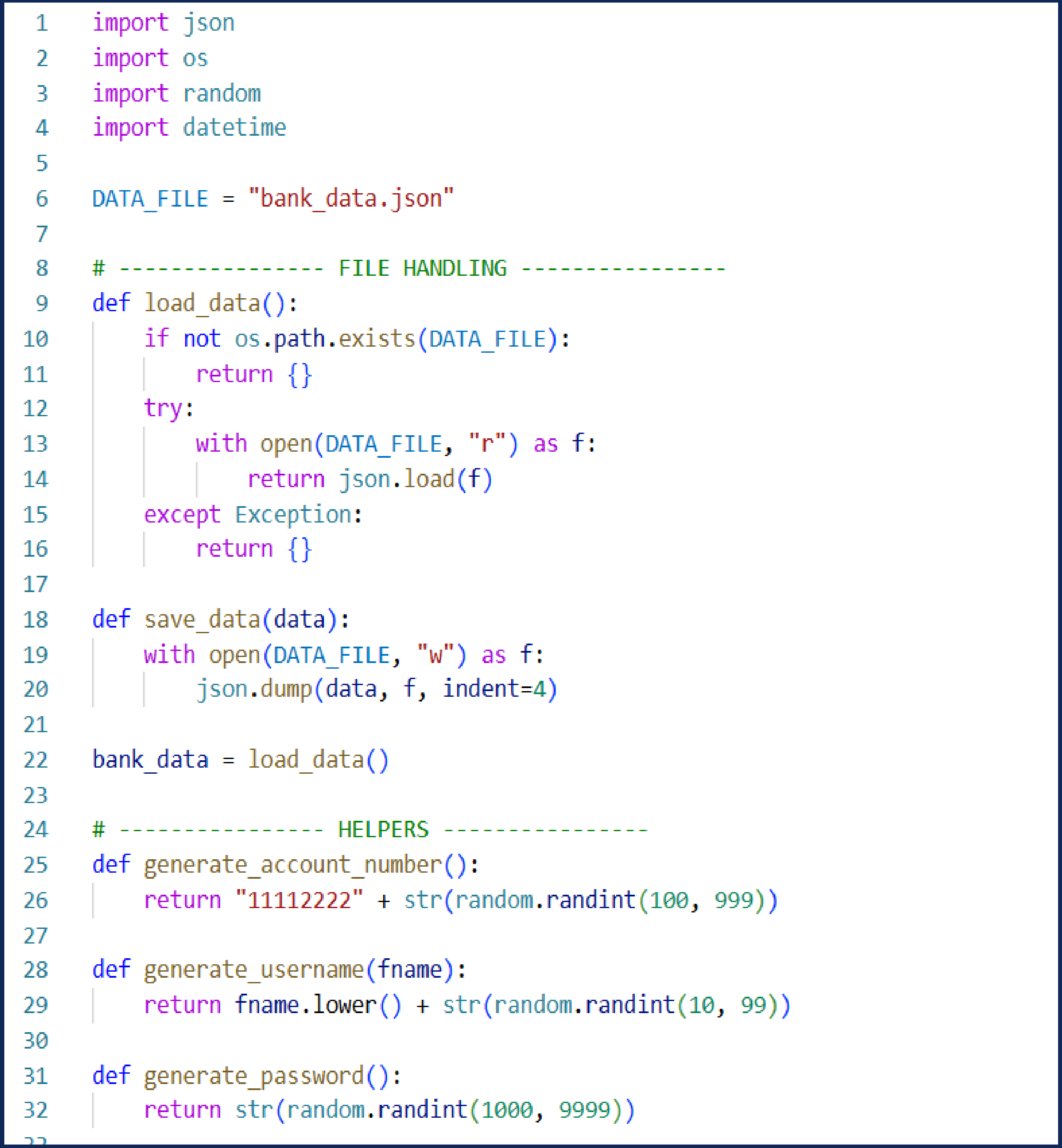
Transaction Recording

Step 5

Account Summary

Importing Libraries & Initial Setup

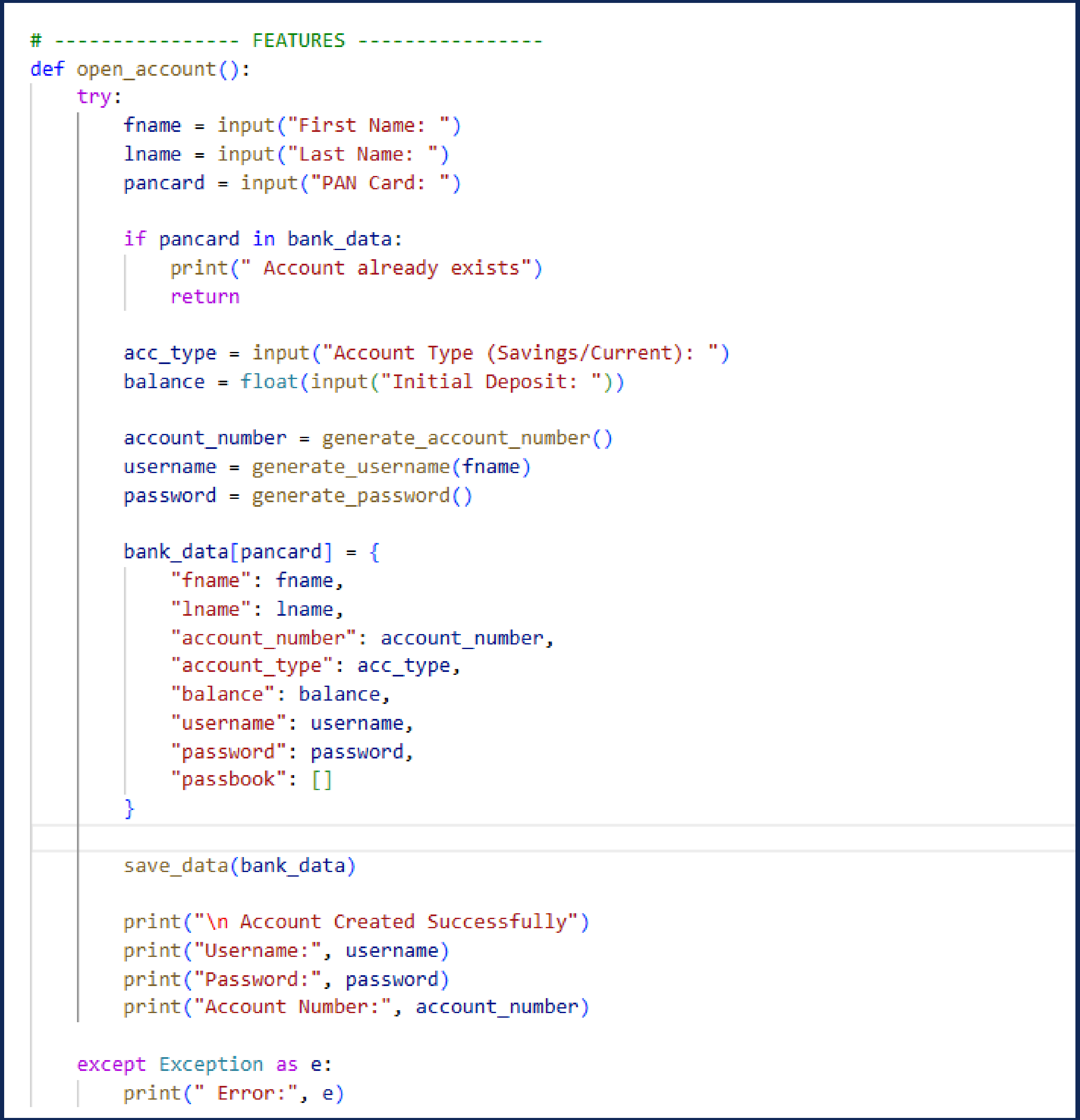
- Imported required Python libraries for system functionality
- json is used for storing and retrieving banking data
- os helps in checking file existence
- random is used to generate account numbers and passwords
- datetime is used to manage transaction date and time
- A single data file is defined to store all banking information permanently.



```python
import json
import os
import random
import datetime

DATA_FILE = "bank_data.json"

# ---------------- FILE HANDLING ----------------
def load_data():
    if not os.path.exists(DATA_FILE):
        return {}
    try:
        with open(DATA_FILE, "r") as f:
            return json.load(f)
    except Exception:
        return {}

def save_data(data):
    with open(DATA_FILE, "w") as f:
        json.dump(data, f, indent=4)

bank_data = load_data()

# ---------------- HELPERS ----------------
def generate_account_number():
    return "11112222" + str(random.randint(100, 999))

def generate_username(fname):
    return fname.lower() + str(random.randint(10, 99))

def generate_password():
    return str(random.randint(1000, 9999))
```

Account Creation Module

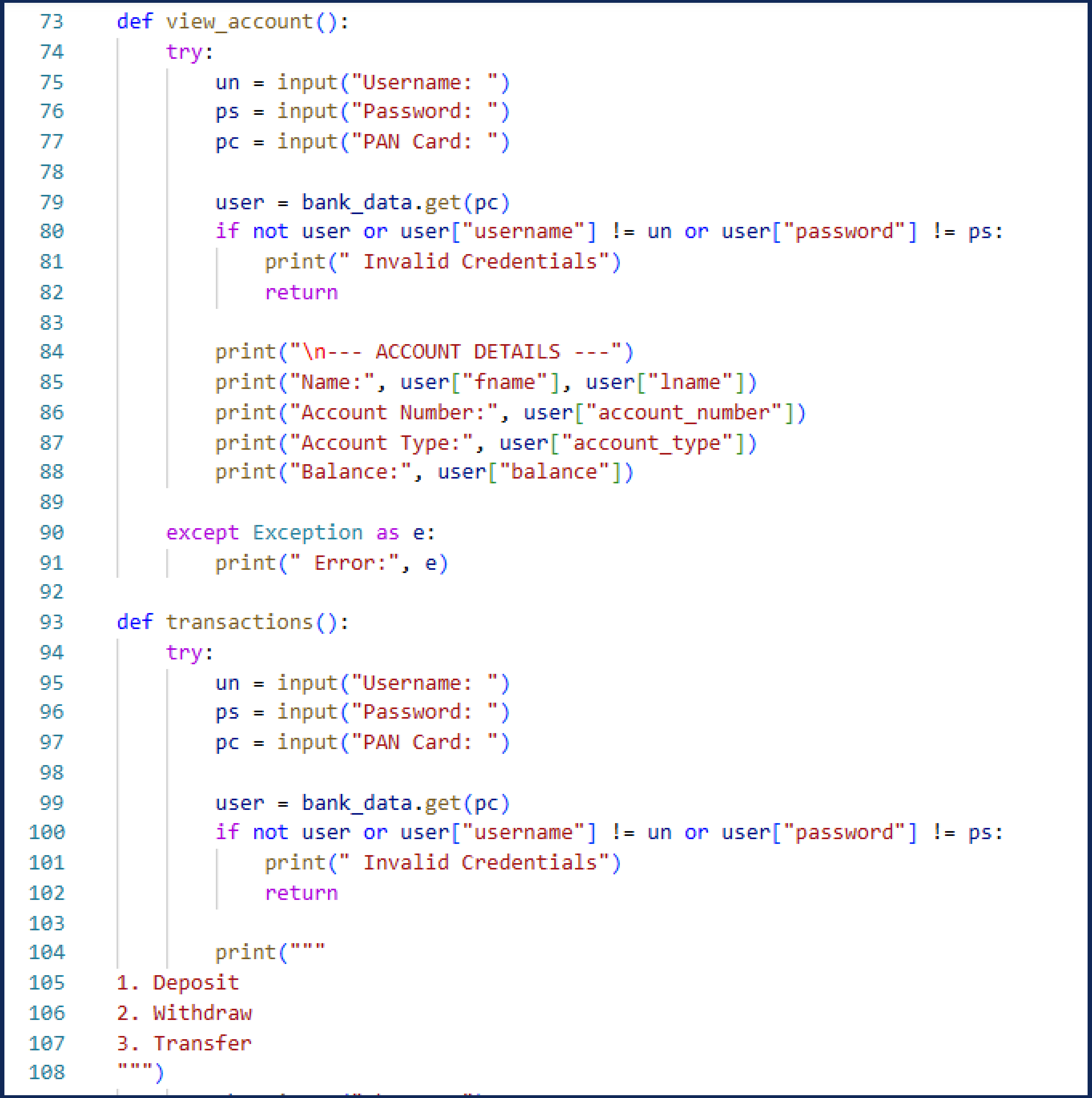
- This module allows users to open a new bank account
- User details like **name, PAN card, and account type** are collected
- System checks whether the account already exists
- A unique **account number, username, and password** are generated
- Initial deposit amount is added to the account
- Error handling is used to avoid program crashes



```python
# ---------------- FEATURES ----------------
def open_account():
    try:
        fname = input("First Name: ")
        lname = input("Last Name: ")
        pancard = input("PAN Card: ")

        if pancard in bank_data:
            print(" Account already exists")
            return

        acc_type = input("Account Type (Savings/Current): ")
        balance = float(input("Initial Deposit: "))

        account_number = generate_account_number()
        username = generate_username(fname)
        password = generate_password()

        bank_data[pancard] = {
            "fname": fname,
            "lname": lname,
            "account_number": account_number,
            "account_type": acc_type,
            "balance": balance,
            "username": username,
            "password": password,
            "passbook": []
        }

        save_data(bank_data)

        print("\n Account Created Successfully")
        print("Username:", username)
        print("Password:", password)
        print("Account Number:", account_number)

    except Exception as e:
        print(" Error:", e)
```

Account View & Authentication

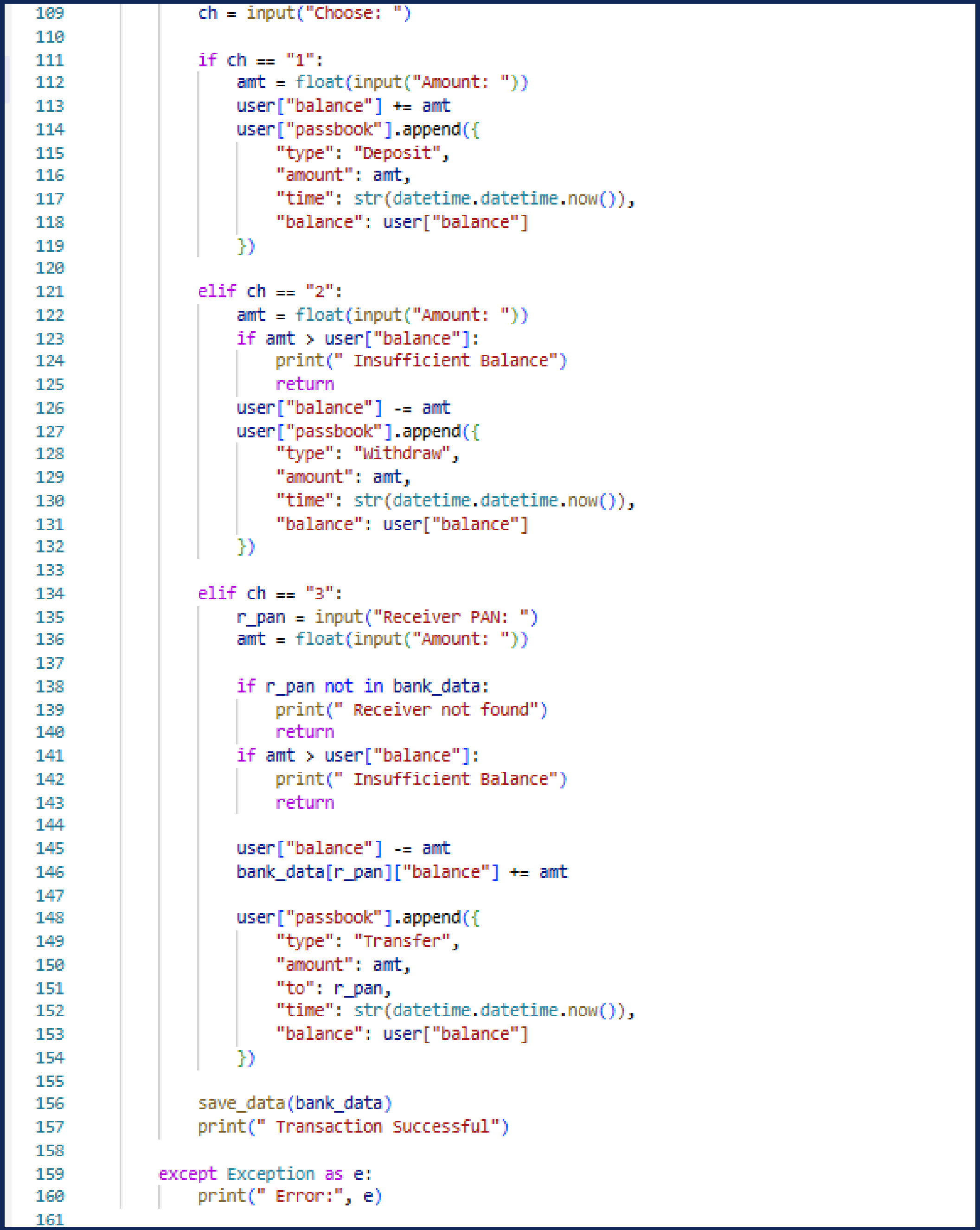
- This module allows users to securely access their bank account
- System verifies credentials from stored data
- If credentials are invalid, access is denied
- On successful authentication, account details are displayed
- Exception handling is used to manage runtime errors



```python
def view_account():
    try:
        un = input("Username: ")
        ps = input("Password: ")
        pc = input("PAN Card: ")

        user = bank_data.get(pc)
        if not user or user["username"] != un or user["password"] != ps:
            print(" Invalid Credentials")
            return

        print("\n--- ACCOUNT DETAILS ---")
        print("Name:", user["fname"], user["lname"])
        print("Account Number:", user["account_number"])
        print("Account Type:", user["account_type"])
        print("Balance:", user["balance"])

    except Exception as e:
        print(" Error:", e)

def transactions():
    try:
        un = input("Username: ")
        ps = input("Password: ")
        pc = input("PAN Card: ")

        user = bank_data.get(pc)
        if not user or user["username"] != un or user["password"] != ps:
            print(" Invalid Credentials")
            return

        print("""
1. Deposit
2. Withdraw
3. Transfer
""")
```

Transaction Management Module

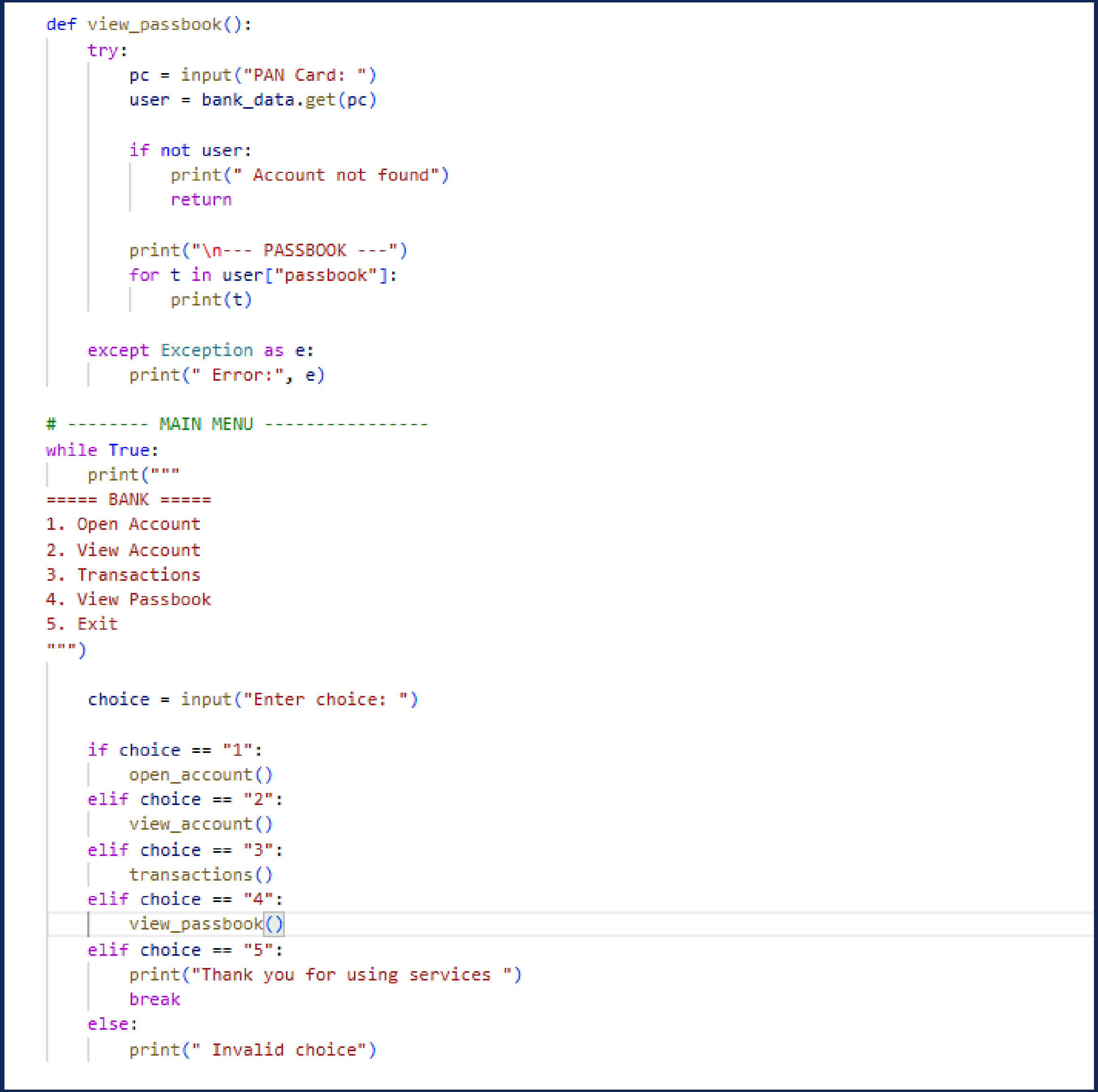
- This module handles all banking transactions.
- User can choose b\w Deposit, Withdraw , and Tranfer
- Deposit increase the account balance
- Withdraw checks for sufficient balance before deduction
- Transfer sends money to another user using PAN card
- Every transaction is recorded in a **passbook** with date and time
- System prevents invalid operations using validations
- Exception handling is used to handle runtime errors



```python
        ch = input("Choose: ")

        if ch == "1":
            amt = float(input("Amount: "))
            user["balance"] += amt
            user["passbook"].append({
                "type": "Deposit",
                "amount": amt,
                "time": str(datetime.datetime.now()),
                "balance": user["balance"]
            })

        elif ch == "2":
            amt = float(input("Amount: "))
            if amt > user["balance"]:
                print(" Insufficient Balance")
                return
            user["balance"] -= amt
            user["passbook"].append({
                "type": "Withdraw",
                "amount": amt,
                "time": str(datetime.datetime.now()),
                "balance": user["balance"]
            })

        elif ch == "3":
            r_pan = input("Receiver PAN: ")
            amt = float(input("Amount: "))

            if r_pan not in bank_data:
                print(" Receiver not found")
                return
            if amt > user["balance"]:
                print(" Insufficient Balance")
                return

            user["balance"] -= amt
            bank_data[r_pan]["balance"] += amt

            user["passbook"].append({
                "type": "Transfer",
                "amount": amt,
                "to": r_pan,
                "time": str(datetime.datetime.now()),
                "balance": user["balance"]
            })

        save_data(bank_data)
        print(" Transaction Successful")

    except Exception as e:
        print(" Error:", e)

```

Passbook & Main Menu

- Displays complete transaction history (passbook)
- Shows date, type, and amount of each transaction
- Verifies account using PAN card
- Menu-driven interface for easy user interaction
- Allows users to perform multiple banking operations
- Handles invalid inputs and errors safely



```python
def view_passbook():
    try:
        pc = input("PAN Card: ")
        user = bank_data.get(pc)

        if not user:
            print(" Account not found")
            return

        print("\n--- PASSBOOK ---")
        for t in user["passbook"]:
            print(t)

    except Exception as e:
        print(" Error:", e)

# -------- MAIN MENU ----------------
while True:
    print("""
===== BANK =====
1. Open Account
2. View Account
3. Transactions
4. View Passbook
5. Exit
""")

    choice = input("Enter choice: ")

    if choice == "1":
        open_account()
    elif choice == "2":
        view_account()
    elif choice == "3":
        transactions()
    elif choice == "4":
        view_passbook()
    elif choice == "5":
        print("Thank you for using services ")
        break
    else:
        print(" Invalid choice")
```

OUTPUT: ACCOUNT CREATION

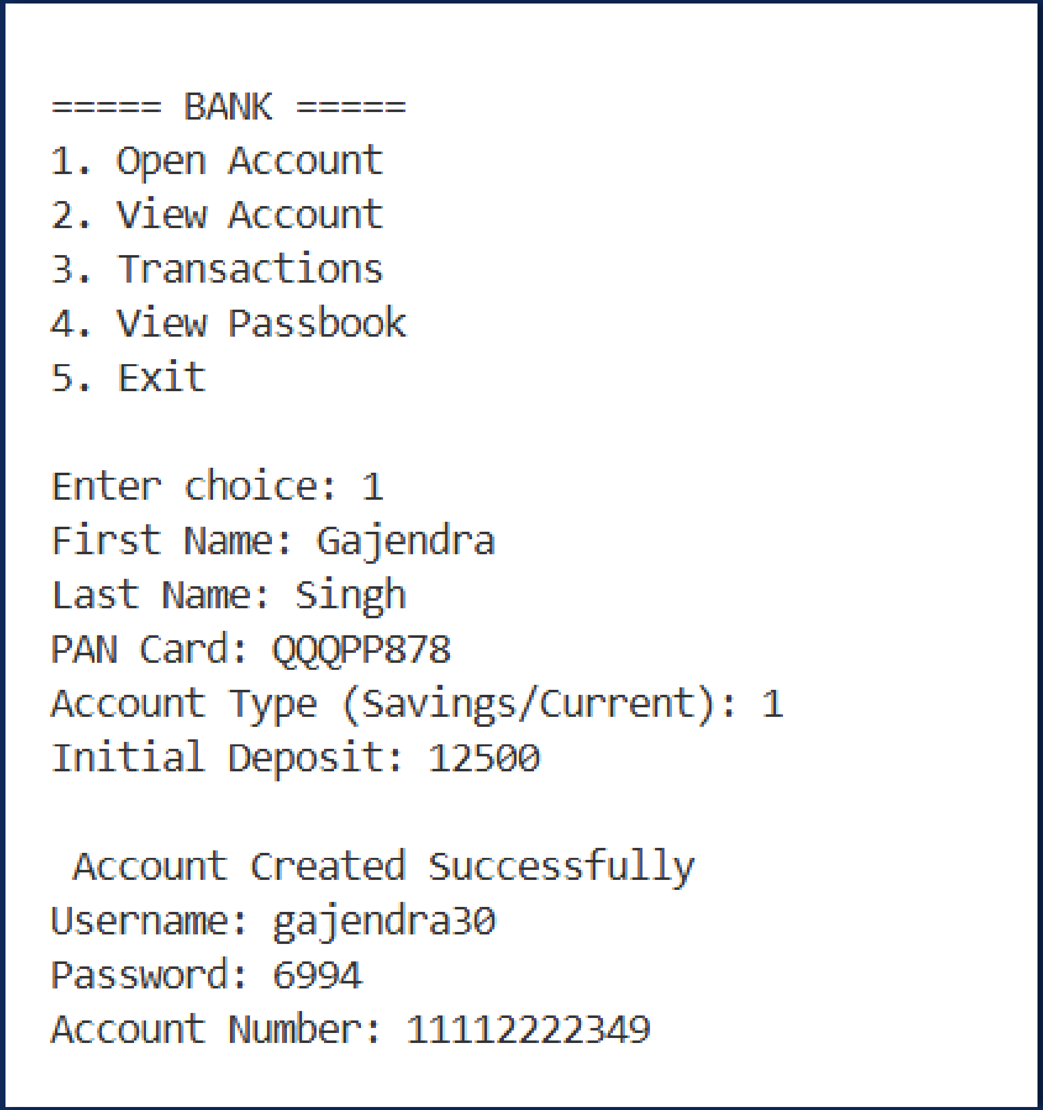
•User selects **Open Account** option from menu

•System collects user details (Name, PAN, Account Type, Deposit)

•Account is created successfully

•System generates **unique Username, Password, and Account Number**

•Initial deposit is added to the account



```
===== BANK =====
1. Open Account
2. View Account
3. Transactions
4. View Passbook
5. Exit

Enter choice: 1
First Name: Gajendra
Last Name: Singh
PAN Card: QQQPP878
Account Type (Savings/Current): 1
Initial Deposit: 12500

 Account Created Successfully
Username: gajendra30
Password: 6994
Account Number: 11112222349
```

DEPOSIT TRANSACTION

•User selects **Transactions** option from main   menu

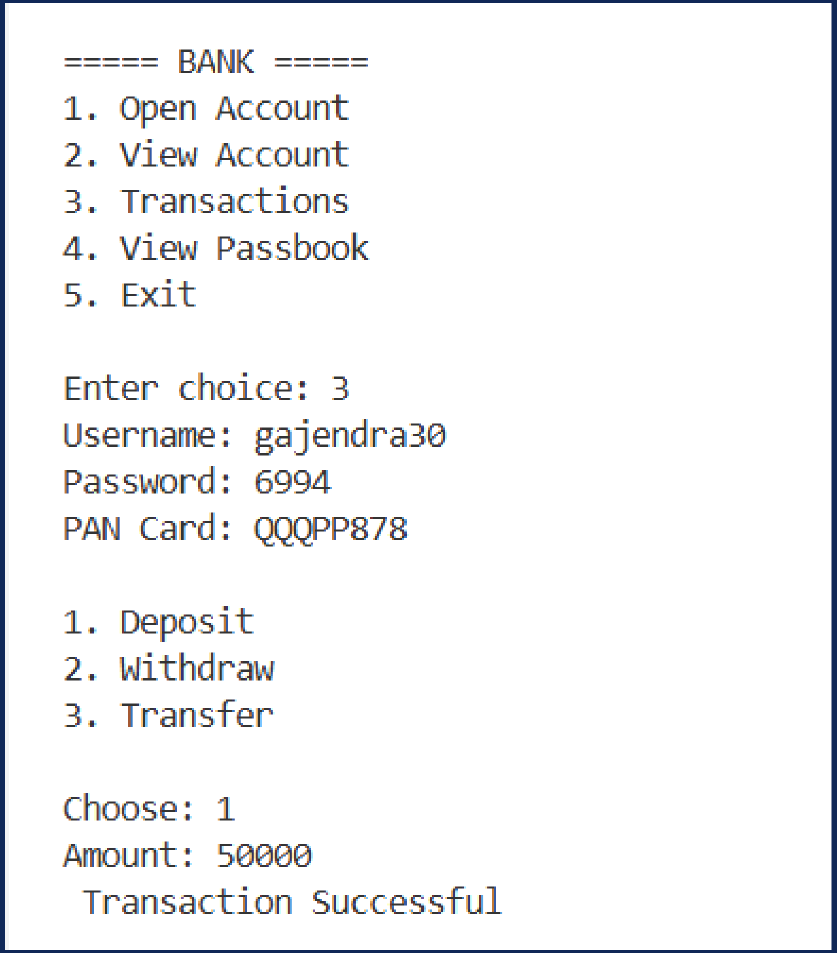
•User login is verified using **Username, Password, and PAN**

•Deposit option is selected

•User enters deposit amount

•Amount is successfully added to account balance

•System confirms **Transaction Successful**



```
===== BANK =====
1. Open Account
2. View Account
3. Transactions
4. View Passbook
5. Exit

Enter choice: 3
Username: gajendra30
Password: 6994
PAN Card: QQQPP878

1. Deposit
2. Withdraw
3. Transfer

Choose: 1
Amount: 50000
 Transaction Successful
```

VIEW PASSBOOK

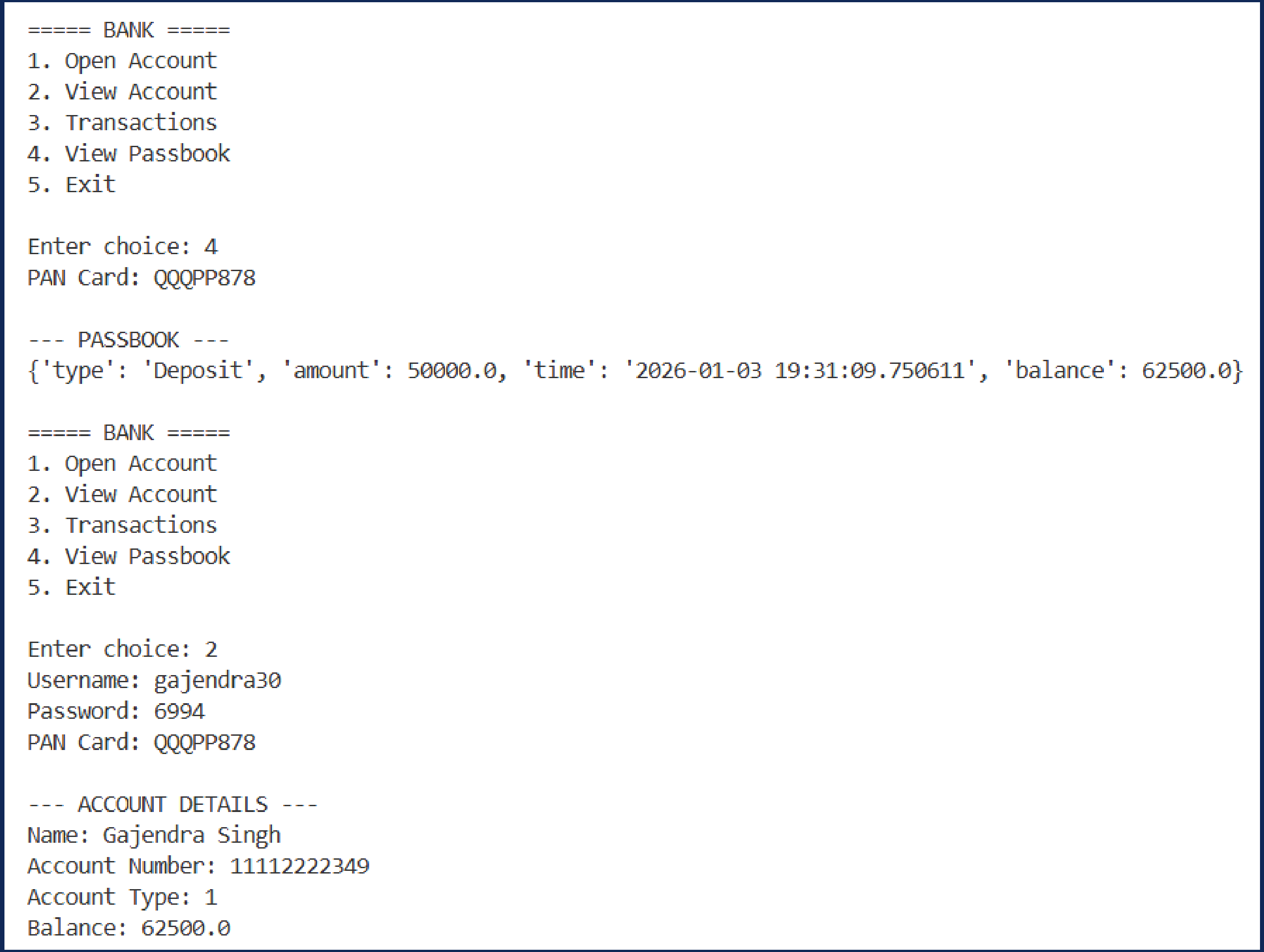
•User selects **View Passbook** option

•PAN Card is used to fetch account details

•System displays **complete transaction history**

• Each entry shows:
✓ Transaction type (Deposit)
✓ Amount
✓ Date & Time
✓ Updated balance

•Passbook helps user track all banking activities



```
===== BANK =====
1. Open Account
2. View Account
3. Transactions
4. View Passbook
5. Exit

Enter choice: 4
PAN Card: QQQPP878

--- PASSBOOK ---
{'type': 'Deposit', 'amount': 50000.0, 'time': '2026-01-03 19:31:09.750611', 'balance': 62500.0}

===== BANK =====
1. Open Account
2. View Account
3. Transactions
4. View Passbook
5. Exit

Enter choice: 2
Username: gajendra30
Password: 6994
PAN Card: QQQPP878

--- ACCOUNT DETAILS ---
Name: Gajendra Singh
Account Number: 11112222349
Account Type: 1
Balance: 62500.0
```

CONCLUSION

This project demonstrates a simple and efficient Bank Account Management System using Python. It allows users to create accounts, perform transactions, and manage their banking activities securely. By using file handling and error handling, the system ensures data safety and smooth execution. Overall, this project helped in understanding real-world application of Python concepts and improved problem-solving skills.

THANK YOU

SUBMITTED BY : Gajendra Singh

GITHUB LINK