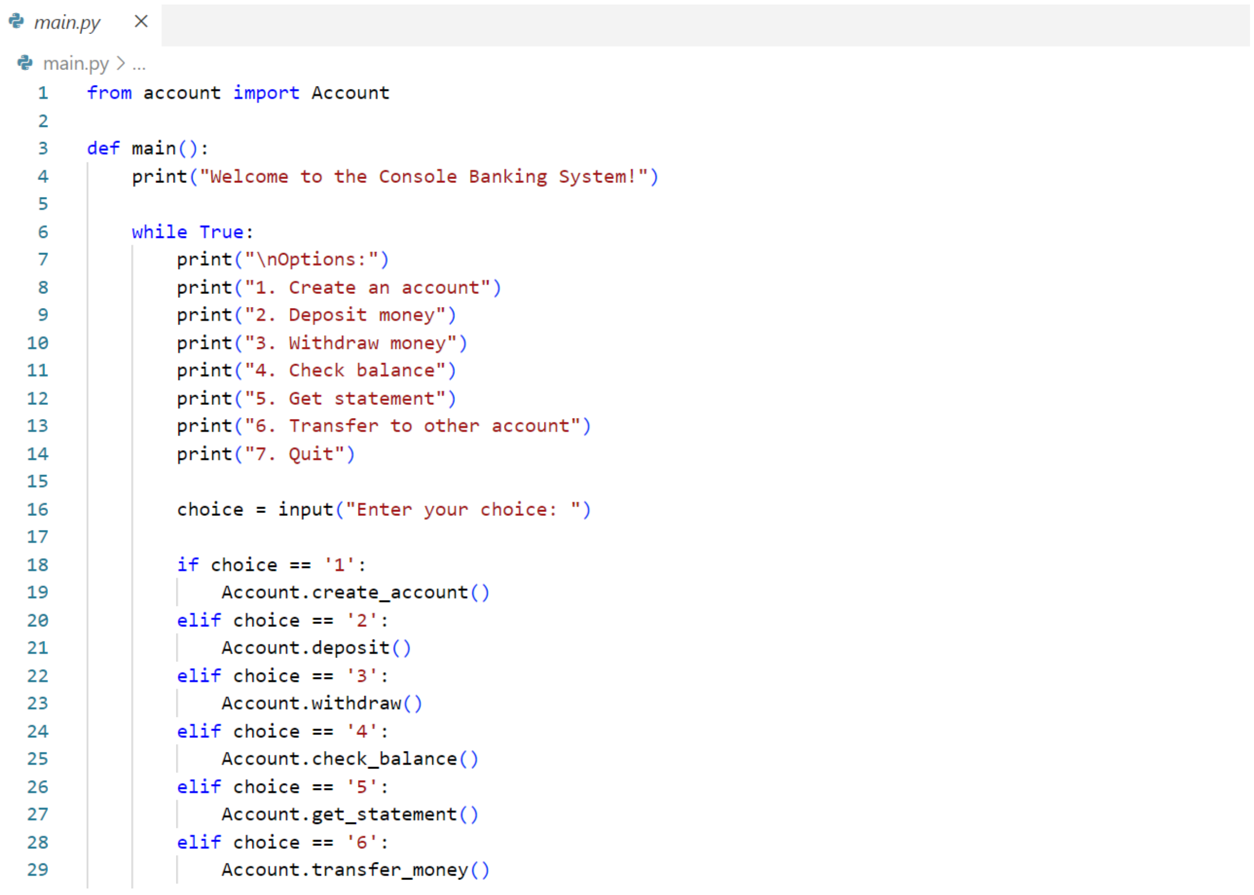
Console based Banking Application

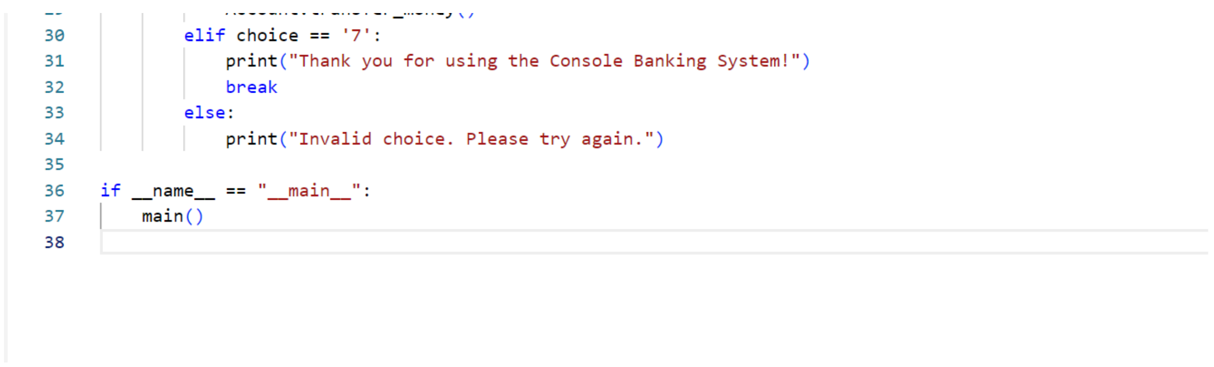
Main.py



main.py

main.py > ...

```python
from account import Account

def main():
    print("Welcome to the Console Banking System!")

    while True:
        print("\nOptions:")
        print("1. Create an account")
        print("2. Deposit money")
        print("3. Withdraw money")
        print("4. Check balance")
        print("5. Get statement")
        print("6. Transfer to other account")
        print("7. Quit")

        choice = input("Enter your choice: ")

        if choice == '1':
            Account.create_account()
        elif choice == '2':
            Account.deposit()
        elif choice == '3':
            Account.withdraw()
        elif choice == '4':
            Account.check_balance()
        elif choice == '5':
            Account.get_statement()
        elif choice == '6':
            Account.transfer_money()
```



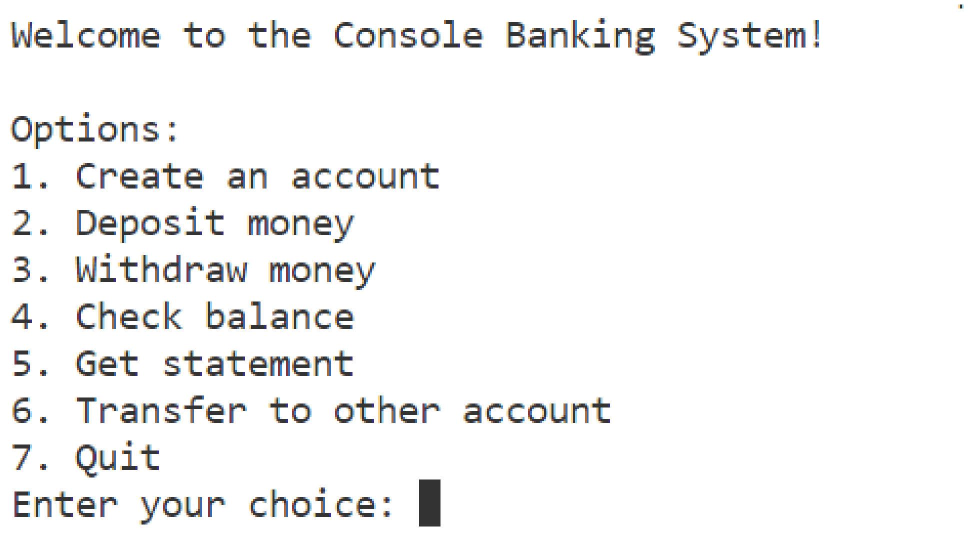
```python
        elif choice == '7':
            print("Thank you for using the Console Banking System!")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Account.py

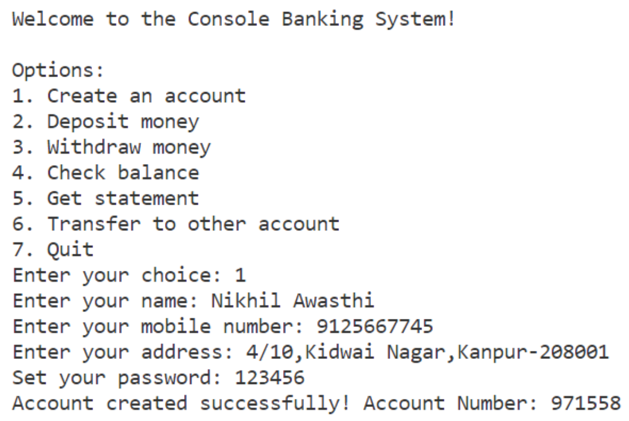
```python
import sqlite3
import random

class Account:
    db_connection = sqlite3.connect('banking_system.db')
    db_cursor = db_connection.cursor()

    def __init__(self, account_number, name, mobile_number, address, password, 
balance=0):
        self.account_number = account_number
        self.name = name
        self.mobile_number = mobile_number
        self.address = address
        self.password = password
        self.balance = balance

        # Create a table if it doesn't exist
        self.db_cursor.execute('''
            CREATE TABLE IF NOT EXISTS transactions (
                transaction_id INTEGER PRIMARY KEY AUTOINCREMENT,
                account_number INTEGER,
                transaction_type TEXT,
                amount REAL,
                transaction_date TEXT,
                FOREIGN KEY (account_number) REFERENCES accounts(account_number)
            )
        ''')
        self.db_connection.commit()

        # Create a table if it doesn't exist
        self.db_cursor.execute('''
            CREATE TABLE IF NOT EXISTS accounts (
                account_number INTEGER PRIMARY KEY,
                name TEXT,
                mobile_number TEXT,
                address TEXT,
                password TEXT,
                balance REAL
            )
        ''')
        self.db_connection.commit()

        # Insert account into the database
        self.db_cursor.execute('INSERT INTO accounts VALUES (?, ?, ?, ?, ?, ?)',
                               (account_number, name, mobile_number, address,
password, balance))
        self.db_connection.commit()

    @classmethod
    def generate_account_number(cls):
        # Generate a unique 6-digit account number
        return random.randint(100000, 999999)

    @classmethod
    def verify_password(cls, account_number, password):
        cls.db_cursor.execute('''
            SELECT password
            FROM accounts
            WHERE account_number = ?
        ''', (account_number,))
        result = cls.db_cursor.fetchone()
        if result and result[0] == password:
            return True
        else:
            return False

    @classmethod
    def log_transaction(cls, account_number, transaction_type, amount):
        cls.db_cursor.execute('''
            INSERT INTO transactions (account_number, transaction_type, amount,
transaction_date)
            VALUES (?, ?, ?, datetime('now'))
        ''', (account_number, transaction_type, amount))
        cls.db_connection.commit()

    @classmethod
    def create_account(cls):
        name = input("Enter your name: ")
        mobile_number = input("Enter your mobile number: ")
        address = input("Enter your address: ")
        password = input("Set your password: ")

        account_number = cls.generate_account_number()
        new_account = cls(account_number, name, mobile_number, address, password)
        print(f"Account created successfully! Account Number:
{new_account.account_number}")

    @classmethod
```

```python
def deposit(cls):
    account_number = int(input("Enter your account number: "))
    password = input("Enter your password: ")

    if cls.verify_password(account_number, password):
        amount = float(input("Enter the amount to deposit: "))
        cls.db_cursor.execute('''
            UPDATE accounts
            SET balance = balance + ?
            WHERE account_number = ?
        ''', (amount, account_number))
        cls.db_connection.commit()
        cls.log_transaction(account_number, 'Deposit', amount)
        print("Deposit successful!")
    else:
        print("Incorrect password or account not found!")

@classmethod
def withdraw(cls):
    account_number = int(input("Enter your account number: "))
    password = input("Enter your password: ")

    if cls.verify_password(account_number, password):
        amount = float(input("Enter the amount to withdraw: "))
        cls.db_cursor.execute('''
            SELECT balance
            FROM accounts
            WHERE account_number = ?
        ''', (account_number,))
        result = cls.db_cursor.fetchone()
        if result:
            balance = result[0]
            if balance >= amount:
                cls.db_cursor.execute('''
                    UPDATE accounts
                    SET balance = balance - ?
                    WHERE account_number = ?
                ''', (amount, account_number))
                cls.db_connection.commit()
                cls.log_transaction(account_number, 'Withdrawal', amount)
                print("Withdrawal successful!")
            else:
                print("Insufficient funds.")
        else:
            print("Account not found!")
        else:
            print("Incorrect password or account not found!")

    @classmethod
    def check_balance(cls):
        account_number = int(input("Enter your account number: "))
        password = input("Enter your password: ")

        if cls.verify_password(account_number, password):
            cls.db_cursor.execute('''
                SELECT balance
                FROM accounts
                WHERE account_number = ?
            ''', (account_number,))
            result = cls.db_cursor.fetchone()
            if result:
                balance = result[0]
                print(f"Account Balance: {balance}")
            else:
                print("Account not found!")
        else:
            print("Incorrect password or account not found!")

    @classmethod
    def get_statement(cls):
        account_number = int(input("Enter your account number: "))
        password = input("Enter your password: ")

        if cls.verify_password(account_number, password):
            cls.db_cursor.execute('''
                SELECT account_number, name, mobile_number, address, balance
                FROM accounts
                WHERE account_number = ?
            ''', (account_number,))
            account_details = cls.db_cursor.fetchone()

            if account_details:
                cls.db_cursor.execute('''
                    SELECT transaction_id, transaction_type, amount,
transaction_date
                    FROM transactions
                    WHERE account_number = ?
                ''', (account_number,))
                transactions = cls.db_cursor.fetchall()
                print("\nAccount Details:")
                print("Account Number\tName\t\tMobile 
Number\tAddress\t\tBalance")
                print("--------------\t----\t\t--------------\t-------\t\t-------
")
                print(f"{account_details[0]}\t\t{account_details[1]}\t\t{account_
details[2]}\t\t{account_details[3]}\t\t{account_details[4]}")

                if transactions:
                    print("\nTransaction Statement:")
                    print("Transaction ID\tType\t\tAmount\t\tDate")
                    print("--------------\t-----------\t-----------\t------------
-------")
                    for transaction in transactions:
                        print(f"{transaction[0]}\t\t{transaction[1]}\t\t{transact
ion[2]}\t\t{transaction[3]}")
                else:
                    print("No transactions found.")
            else:
                print("Account not found.")
        else:
            print("Incorrect password or account not found!")

    @classmethod
    def transfer_money(cls):
        from_account_number = int(input("Enter your account number: "))
        from_password = input("Enter your password: ")

        if cls.verify_password(from_account_number, from_password):
            to_account_number = int(input("Enter the recipient's account number: 
"))
            amount = float(input("Enter the amount to transfer: "))

            cls.db_cursor.execute('''
                SELECT balance
                FROM accounts
                WHERE account_number = ?
            ''', (from_account_number,))
            from_balance = cls.db_cursor.fetchone()

            if from_balance and from_balance[0] >= amount:
                cls.db_cursor.execute('''
                    UPDATE accounts
                    SET balance = balance - ?
                    WHERE account_number = ?
                ''', (amount, from_account_number))

                cls.db_cursor.execute('''
                    UPDATE accounts
                    SET balance = balance + ?
                    WHERE account_number = ?
                ''', (amount, to_account_number))

                cls.db_connection.commit()
                cls.log_transaction(from_account_number, 'Transfer to ' +
str(to_account_number), amount)
                cls.log_transaction(to_account_number, 'Transfer from ' +
str(from_account_number), amount)

                print("Transfer successful!")
            else:
                print("Insufficient funds or invalid account.")
        else:
            print("Incorrect password or account not found!")
```

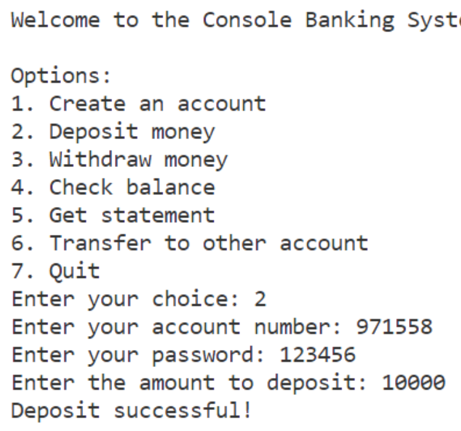
Output :-



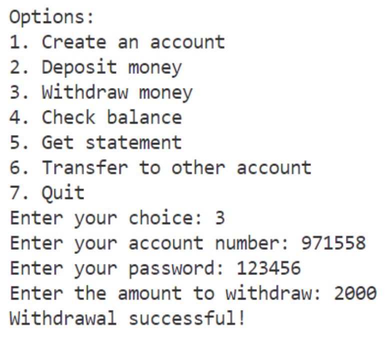
```
Welcome to the Console Banking System!

Options:
1. Create an account
2. Deposit money
3. Withdraw money
4. Check balance
5. Get statement
6. Transfer to other account
7. Quit
Enter your choice: 
```



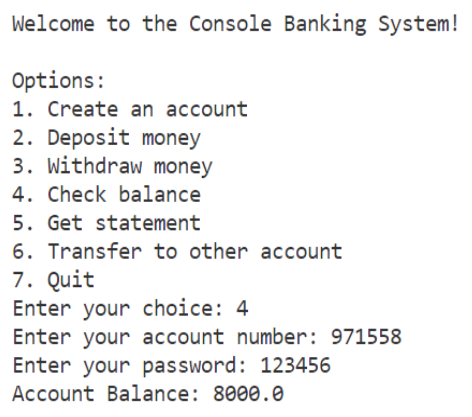
```
Welcome to the Console Banking System!

Options:
1. Create an account
2. Deposit money
3. Withdraw money
4. Check balance
5. Get statement
6. Transfer to other account
7. Quit
Enter your choice: 1
Enter your name: Nikhil Awasthi
Enter your mobile number: 9125667745
Enter your address: 4/10,Kidwai Nagar,Kanpur-208001
Set your password: 123456
Account created successfully! Account Number: 971558
```



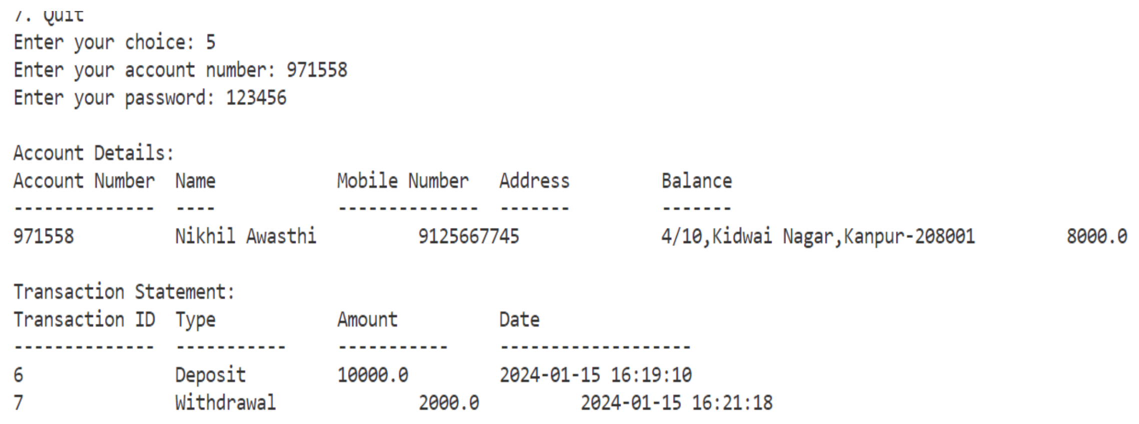
```
Welcome to the Console Banking System!

Options:
1. Create an account
2. Deposit money
3. Withdraw money
4. Check balance
5. Get statement
6. Transfer to other account
7. Quit
Enter your choice: 2
Enter your account number: 971558
Enter your password: 123456
Enter the amount to deposit: 10000
Deposit successful!
```



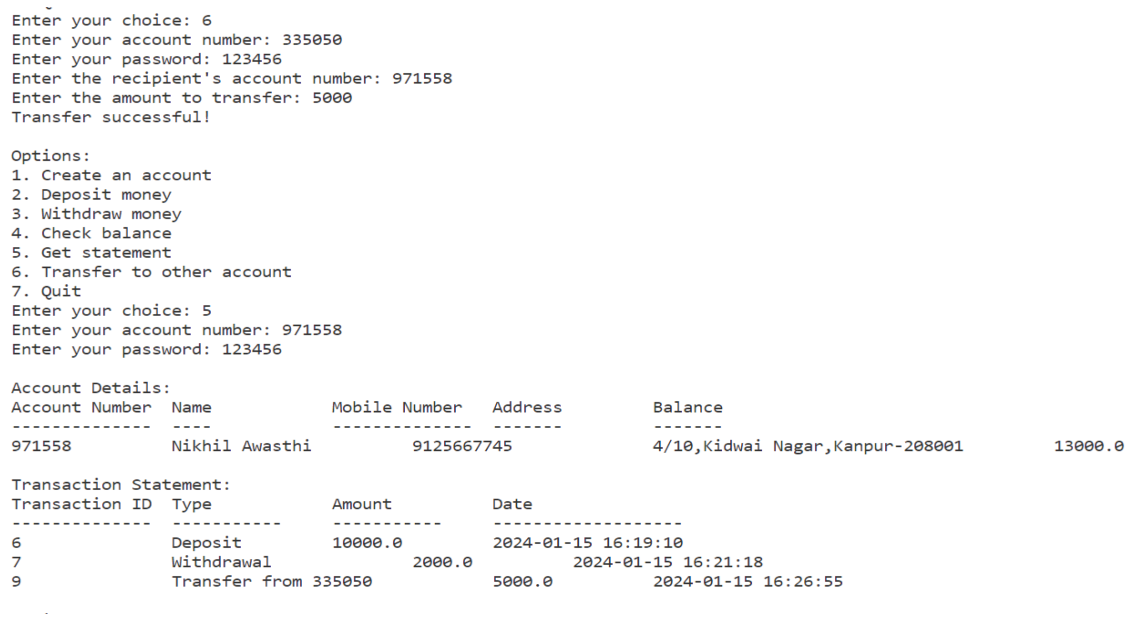
```
Options:
1. Create an account
2. Deposit money
3. Withdraw money
4. Check balance
5. Get statement
6. Transfer to other account
7. Quit
Enter your choice: 3
Enter your account number: 971558
Enter your password: 123456
Enter the amount to withdraw: 2000
Withdrawal successful!
```



```
Welcome to the Console Banking System!

Options:
1. Create an account
2. Deposit money
3. Withdraw money
4. Check balance
5. Get statement
6. Transfer to other account
7. Quit
Enter your choice: 4
Enter your account number: 971558
Enter your password: 123456
Account Balance: 8000.0
```

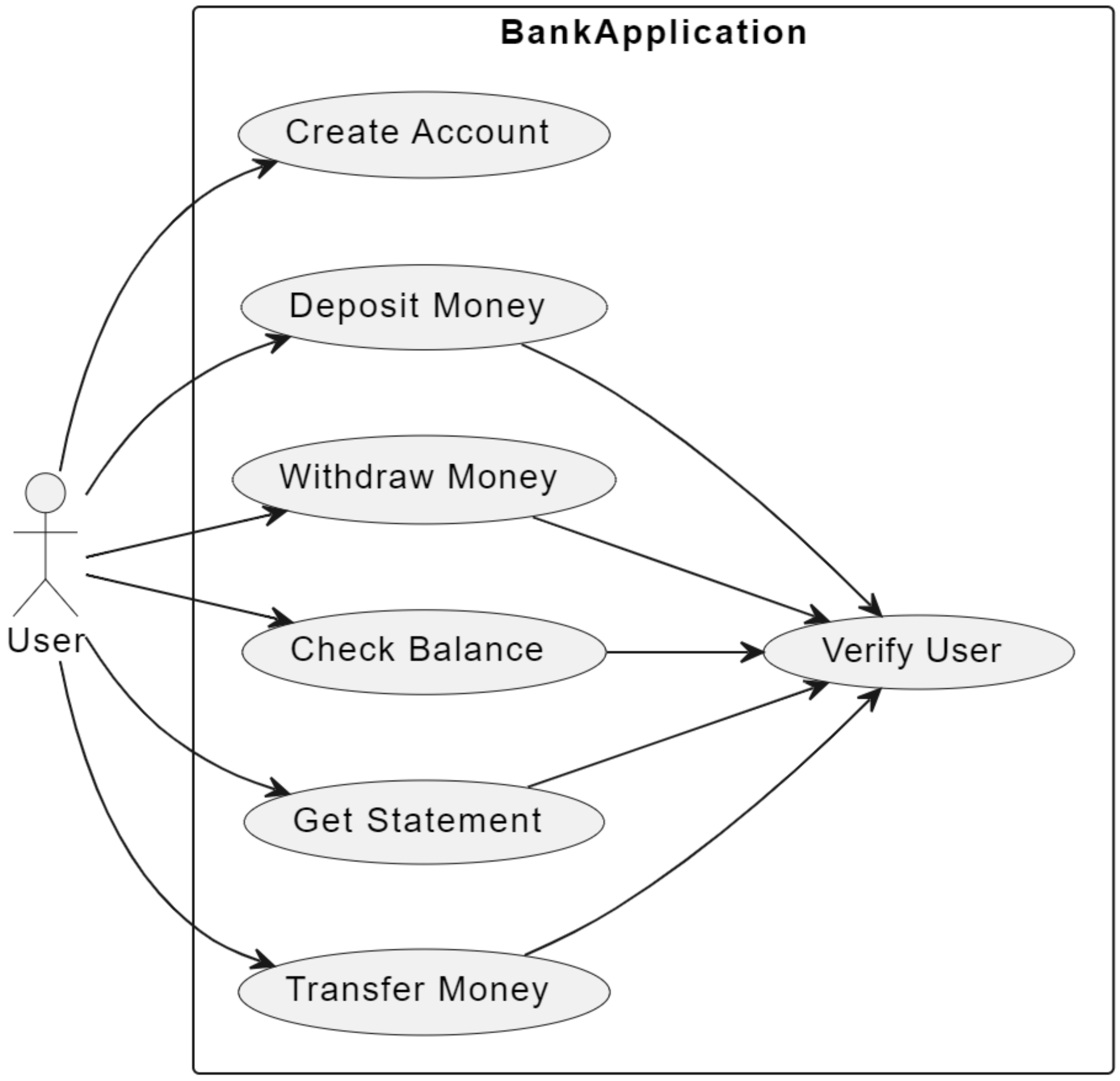


```
7. Quit
Enter your choice: 5
Enter your account number: 971558
Enter your password: 123456

Account Details:
Account Number  Name            Mobile Number   Address         Balance
--------------  ----            --------------  -------         -------
971558          Nikhil Awasthi          9125667745              4/10,Kidwai Nagar,Kanpur-208001         8000.0

Transaction Statement:
Transaction ID  Type            Amount          Date
--------------  -----------     -----------     -------------------
6               Deposit         10000.0         2024-01-15 16:19:10
7               Withdrawal              2000.0          2024-01-15 16:21:18
```

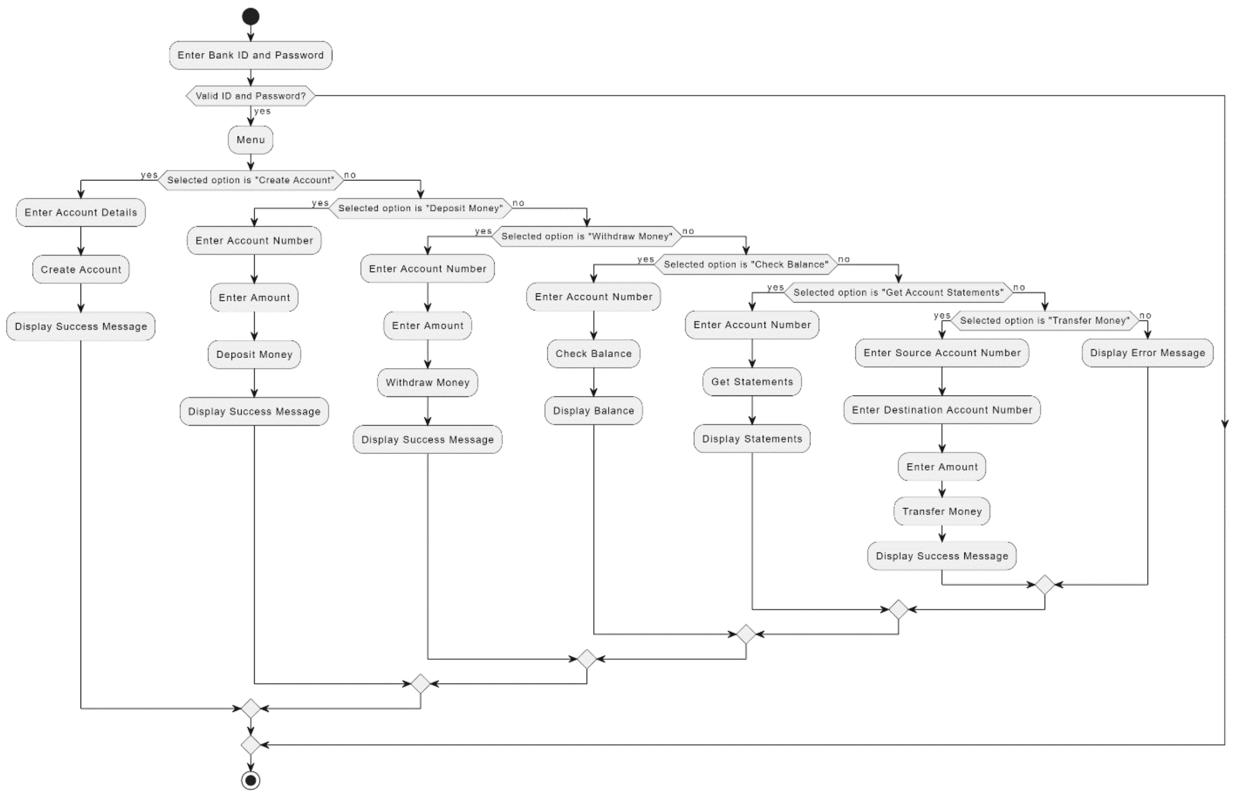


```
Enter your choice: 6
Enter your account number: 335050
Enter your password: 123456
Enter the recipient's account number: 971558
Enter the amount to transfer: 5000
Transfer successful!

Options:
1. Create an account
2. Deposit money
3. Withdraw money
4. Check balance
5. Get statement
6. Transfer to other account
7. Quit
Enter your choice: 5
Enter your account number: 971558
Enter your password: 123456

Account Details:
Account Number  Name            Mobile Number   Address         Balance
--------------  ----            --------------  -------         -------
971558          Nikhil Awasthi          9125667745              4/10,Kidwai Nagar,Kanpur-208001         13000.0

Transaction Statement:
Transaction ID  Type            Amount          Date
--------------  -----------     -----------     -------------------
6               Deposit         10000.0         2024-01-15 16:19:10
7               Withdrawal              2000.0          2024-01-15 16:21:18
9               Transfer from 335050            5000.0          2024-01-15 16:26:55
```

Use Case Diagram



Flow Diagram



Enter Bank ID and Password
Valid ID and Password?
yes
Menu
yes
Selected option is "Create Account"
no
Enter Account Details
Create Account
Display Success Message
yes
Selected option is "Deposit Money"
no
Enter Account Number
Enter Amount
Deposit Money
Display Success Message
yes
Selected option is "Withdraw Money"
no
Enter Account Number
Enter Amount
Withdraw Money
Display Success Message
yes
Selected option is "Check Balance"
no
Enter Account Number
Check Balance
Display Balance
yes
Selected option is "Get Account Statements"
no
Enter Account Number
Get Statements
Display Statements
yes
Selected option is "Transfer Money"
no
Enter Source Account Number
Enter Destination Account Number
Enter Amount
Transfer Money
Display Success Message
Display Error Message