Online Banking System

Student Name: Laith Ismail Ahmed Hijazi

Student ID: 64220039

Email: laith.hijazi@std.medipol.edu.tr

Project Overview

Project Information:

- The Online Banking System is a digital platform designed to simulate core banking functionalities.

- **Purpose:** To provide a user-friendly and secure platform for banking operations such as deposit, withdraw, and transfer.

- **Target Audience:** Administrators and Account Holders.

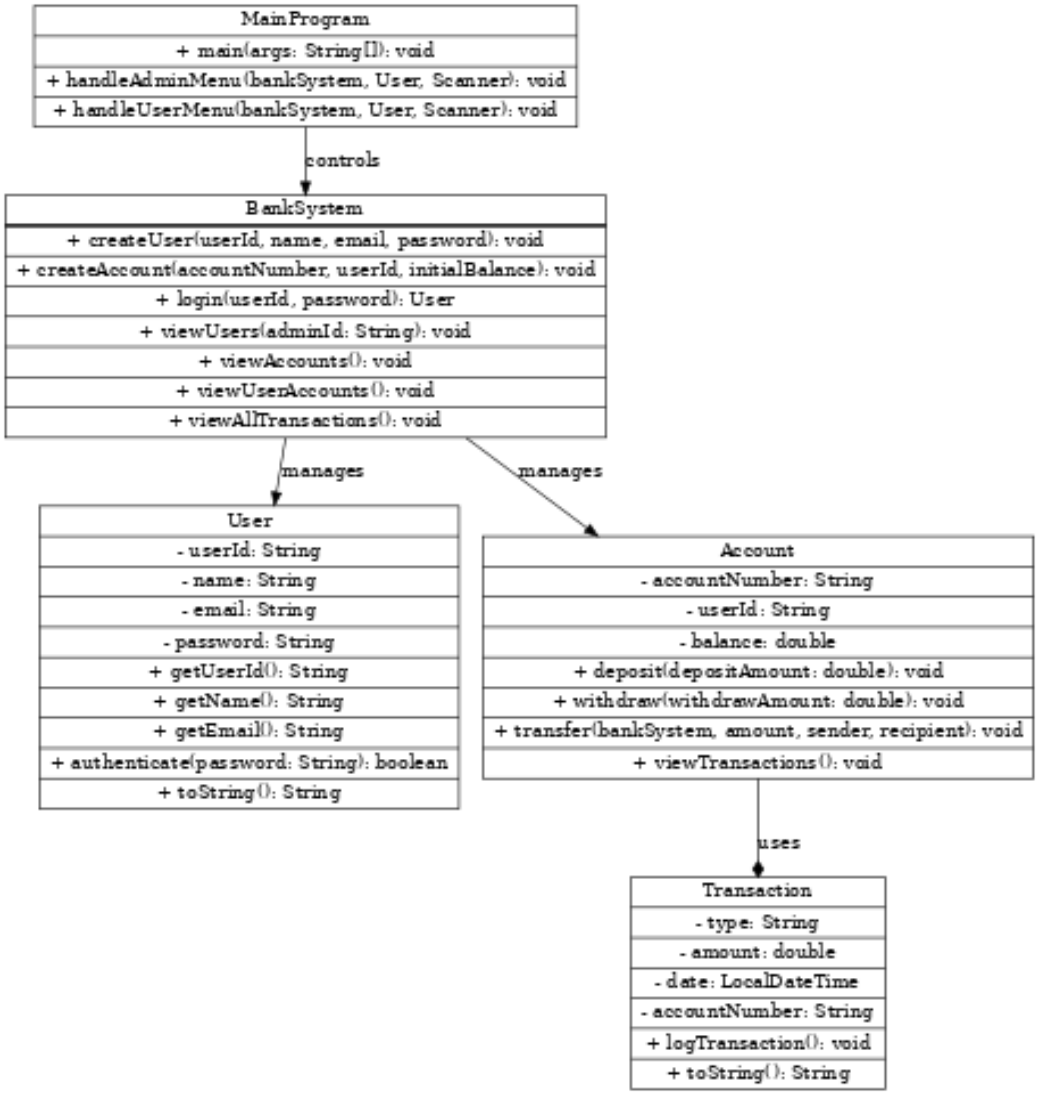
Problem Statement:

- Provides a digital solution for managing bank accounts and performing financial operations.
- Removes the need for physical visits to the bank, allowing users to manage their finances easily and securely online.

System Architecture

Class Diagram:

Illustration of relationships between User, Account, Transaction, and bankSystem classes.



Key Features

1. **User Management:** Admins can create/manage users, and users can securely log in using their credentials.
2. **Account Management:** Users can create accounts linked to their profiles and manage balances.
3. **Financial Transactions:** Deposit money, withdraw, and transfer to other accounts.
4. **Transaction History:** Keeps a detailed record of all transactions, and users can view their transaction history.
5. **Admin Tools:** View registered users, created accounts, and transactions.

OOP Concepts Implemented

1. **Classes and Objects:**

- User class for user details (e.g., name, email, password) and authentication.

- Account class for bank account management.

- Transaction class for logging financial operations.

- BankSystem as the controller managing all interactions.

- mainProgram acts an the entry point handling user input through menus.

2. **Encapsulation:** Private attributes (e.g. userId, balance) with getter/setter methods was used to protect data ensuring it is accessed only through those methods.

3. **Polymorphism:** toString() method was overridden in User, Account, and Transaction classes to provide custom representations.

4. **Abstraction:** There are no abstract classes or interfaces, but abstraction is achieved by hiding implementation details in methods like deposit(), withdraw(), and transfer(), and bankSystem class which acts as the central point for all operations.

Evaluation of OOP Concepts

1. **Classes & Objects:** Well-organized classes. Each class has a clear role, such as User for user details, Account for account management, Transaction for recording transactions, and bankSystem manages the overall system.

2. **Encapsulation**

In user class:

```
public String getUserId(){
    return userId;
}

public String getName(){
    return name;
}

public String getEmail(){
    return email;
}
```

In Account class:

```
public void setAccountNumber(String accountNumber) {
    this.accountNumber = accountNumber;
}

public void setUserId(String userId) {
    this.userId = userId;
}

public void setBalance(double balance) {
    this.balance = balance;
}
```

3. **Polymorphism:** Overridden toString() method in multiple classes for customized representations.

In User Class:

```
@Override
public String toString(){
    return "User [userId=" + userId + ", name=" + name + ", email=" + email + ", password=" + password + "]";
}
```

In Account Class:

```
@Override
public String toString(){
    return "Account [accountNumber=" + accountNumber + ", userId=" + userId + ", balance=" + balance + "]";
}
```

4. **Abstraction:** Simplified methods for user operations to abstract away the internal logic, making it easier for them to interact with the system.

Challenges Faced

1. **Ensuring smooth interaction between classes**

Solution: Used UML diagrams for clarity.

2. **Addressing invalid inputs and edge cases**

Solution: Implemented try-catch blocks and input validation.

3. **Managing transaction logs with file I/O**

Solution: Utilized FileWriter and handled scenarios like missing files.

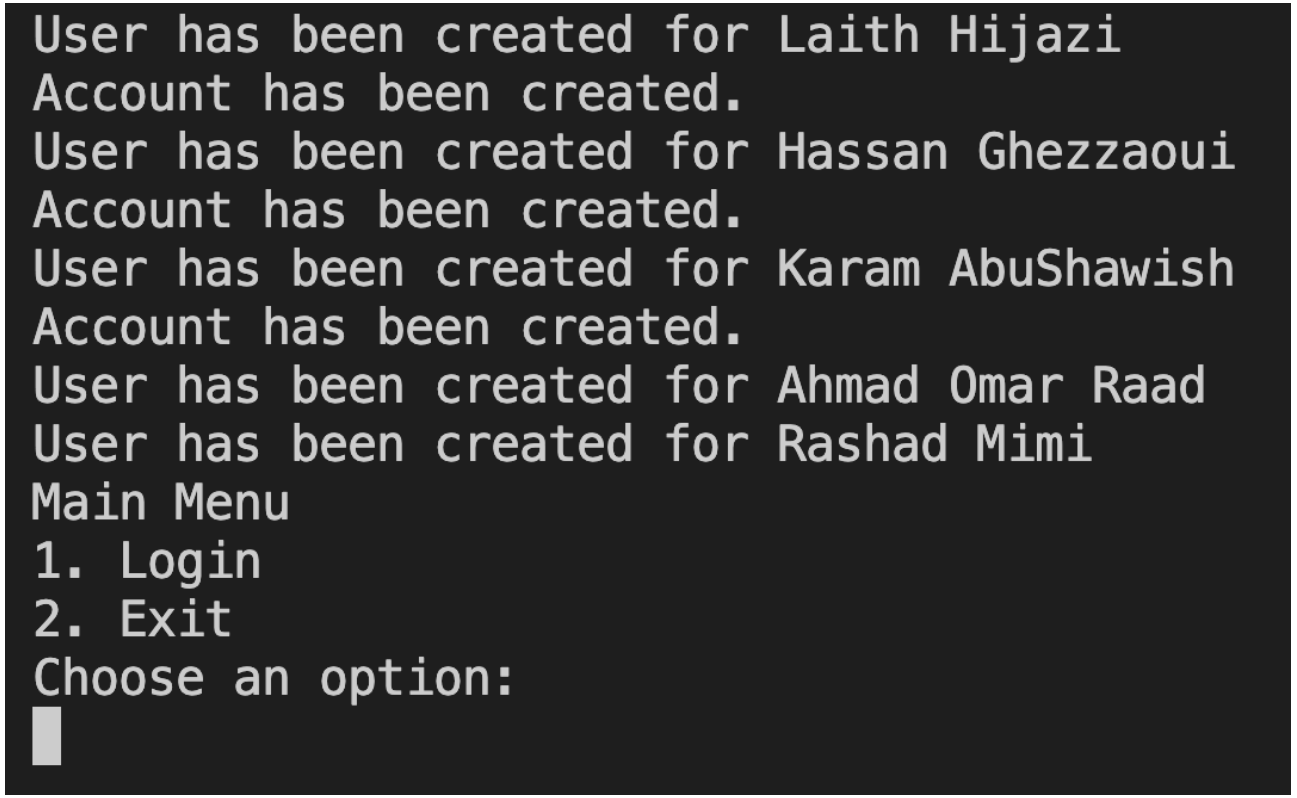
Code Generator Tools

Tool Used: Visual Studio Code's Source Actions Feature

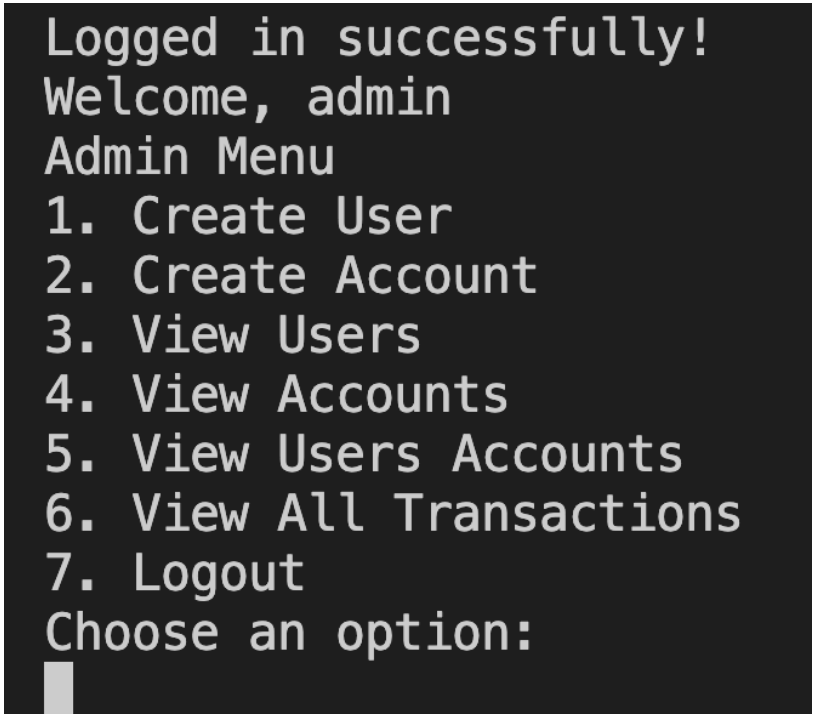
- Automatically generated setters, getters, and constructors.
- Saved development time and improved efficiency.

Project Screenshots

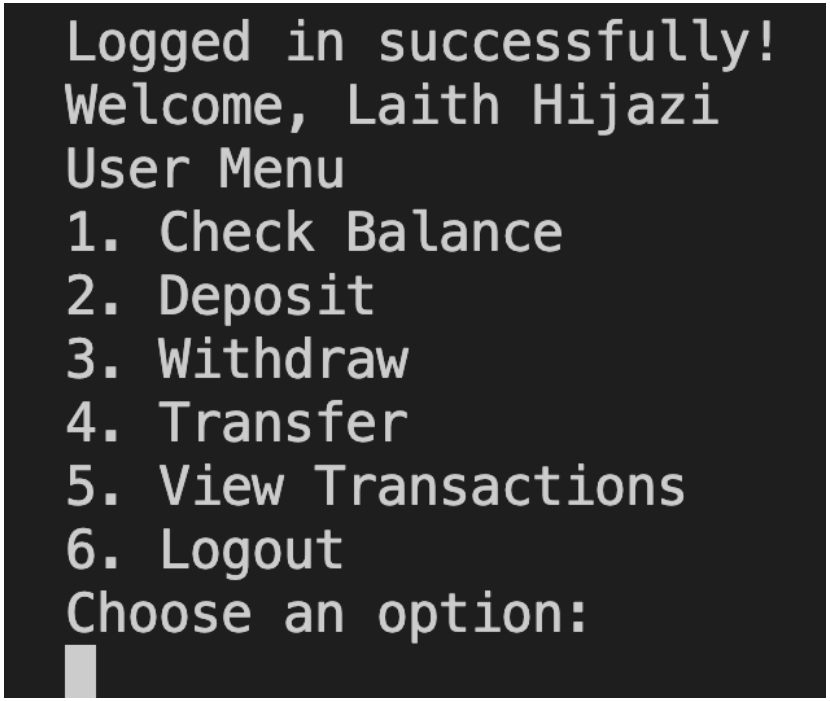
Login Screen: User and admin login interface.



Admin Menu: Options for managing users and accounts.



User Menu: Functions for checking balance, deposits, withdrawals, transfers, and viewing transaction history.



Conclusion

- The project demonstrates effective use of OOP principles.
- Simplifies real-world banking operations through modular design.
- **Future Enhancements:** Integrate a database and improve UI/UX.