Low Level Design (LLD)

Banking Application in Java (Console Based Application)

By

Aniket Narode

Document Version Control

Date Issued	Version	Description	Author
11th March 2023	1.1	First Draft	Aniket Narode

Contents

Abstract

A banking application in Java can be implemented as a console-based application. The purpose of the application is to provide basic banking functionalities such as creating a new account, depositing, withdrawing, and checking account balance. Here's an abstract example of how the application can be structured:

1. Deposit Method: The deposit method will take an amount as input and add it to the account balance and print it to the console

2. Withdraw Method: The withdraw method will take an amount as input and deduct it from the account balance. The method will also check if the account has enough balance to withdraw the amount.

3. Send Money Method: The send money method will take two account objects and an amount as input. The method will first check if the sender account has enough balance to transfer the amount. If yes, it will deduct the amount from the sender account and add it to the receiver account.

1 Introduction

1.1 Why this Low-Level Design Document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2 Scope

Low-level design (LLD) is a component-level design process that follows a step-by step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

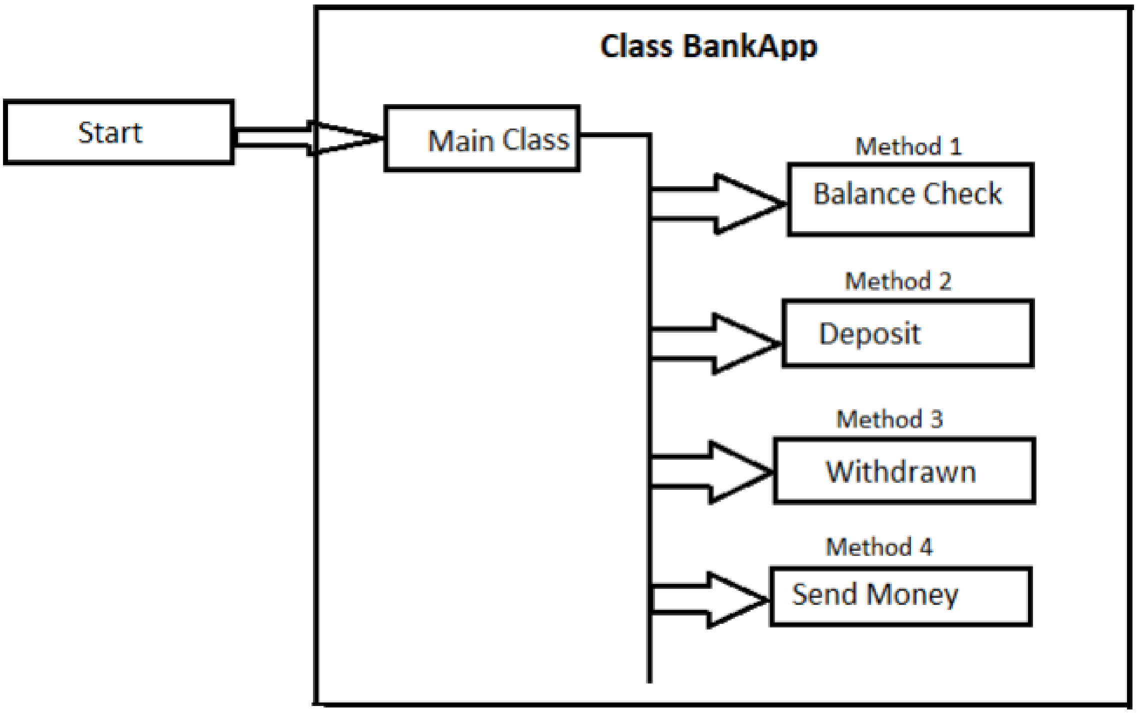
1.3 Constraints

1. Limited User Interface: Console-based applications have limited user interfaces compared to GUI-based applications. This makes it challenging to provide a user-friendly experience and can be challenging for customers who are not familiar with using the command line.

2. Limited Scalability: As the number of customers and transactions increase, console-based applications may become less efficient and more challenging to manage. This may lead to longer wait times for customers, which can negatively impact their banking experience.

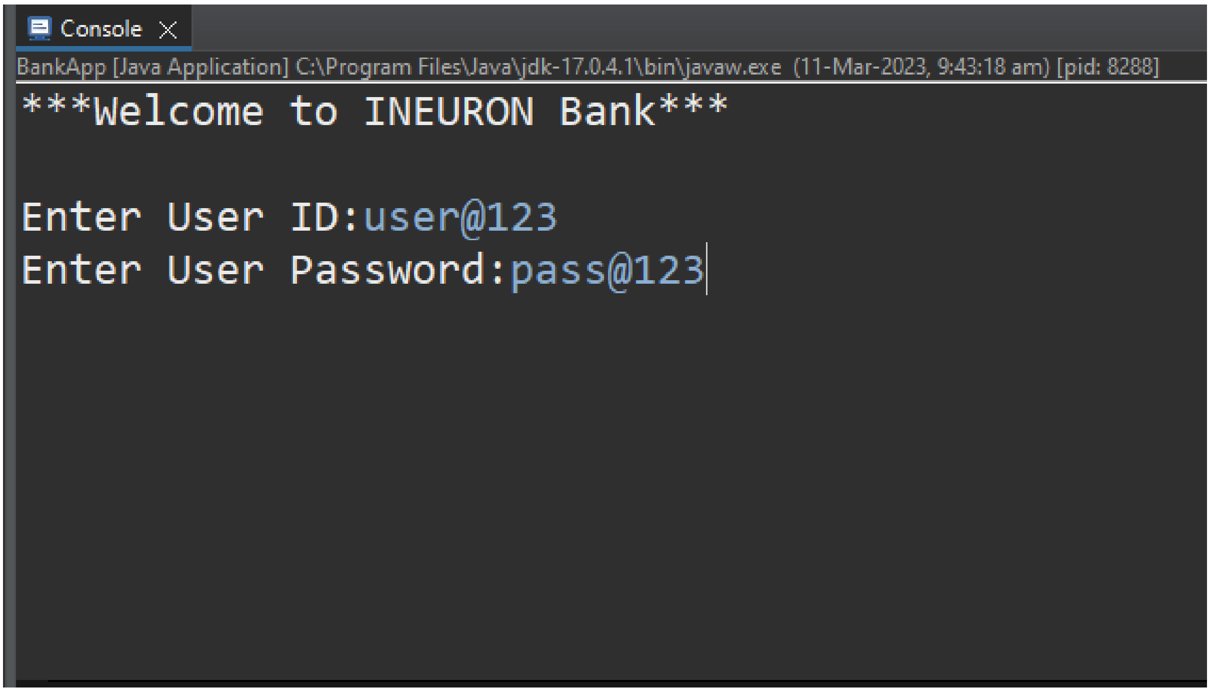
3. Limited Security: Console-based applications may not have the same level of security as GUI-based applications, making them more vulnerable to security breaches. Sensitive customer information, such as account numbers and passwords, may be more susceptible to theft or hacking.

2 Architecture



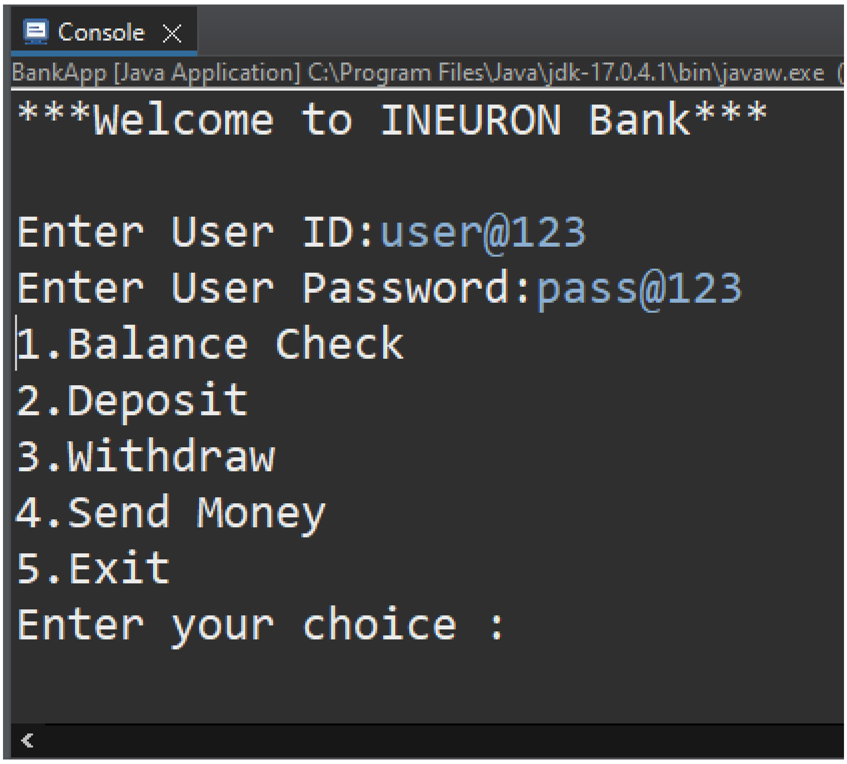
3 User I/O workflow

1. When you run the code In source file in banking Application folder first you will get login page where you have to enter user id and Password to enter in options



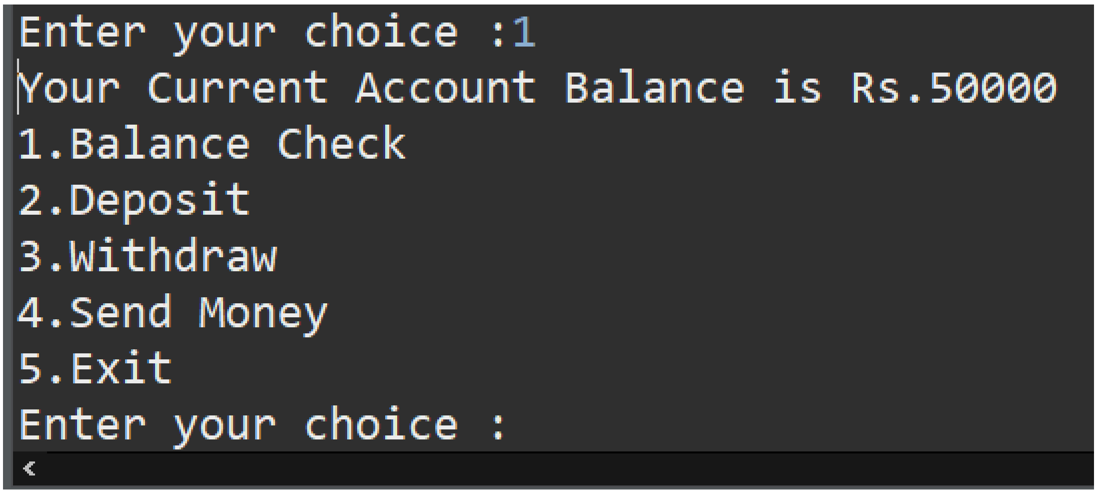
```
BankApp [Java Application] C:\Program Files\Java\jdk-17.0.4.1\bin\javaw.exe  (11-Mar-2023, 9:43:18 am) [pid: 8288]
***Welcome to INEURON Bank***

Enter User ID:user@123
Enter User Password:pass@123
```

2. After entering user id and password user will get option window where user can select options



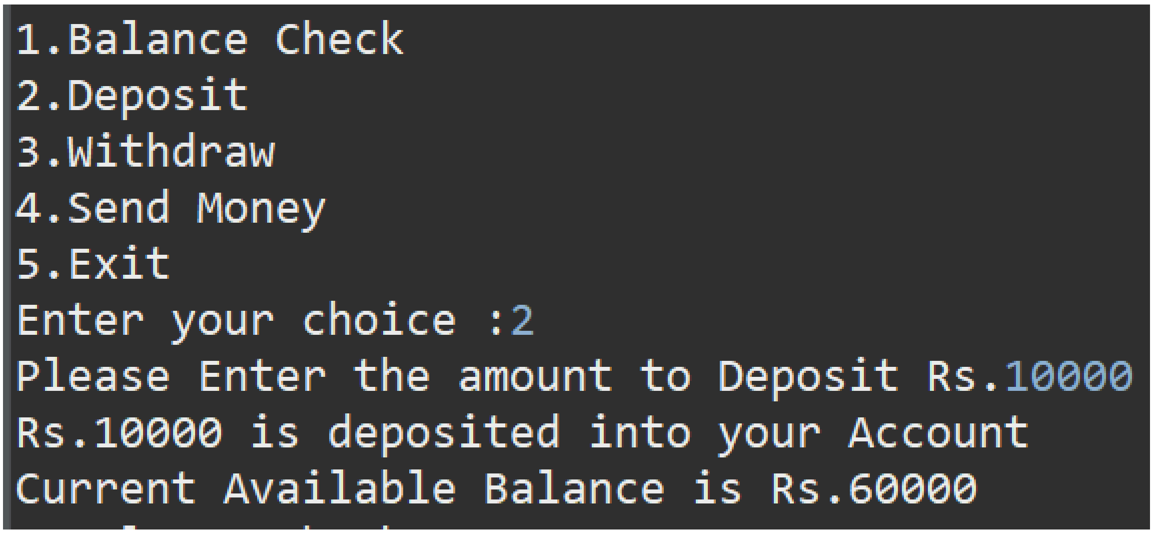
```
BankApp [Java Application] C:\Program Files\Java\jdk-17.0.4.1\bin\javaw.exe  (
***Welcome to INEURON Bank***

Enter User ID:user@123
Enter User Password:pass@123
1.Balance Check
2.Deposit
3.Withdraw
4.Send Money
5.Exit
Enter your choice :
```

3. When user select option 1 for balance check main method will call the check method which will return amount and print the current balance amount



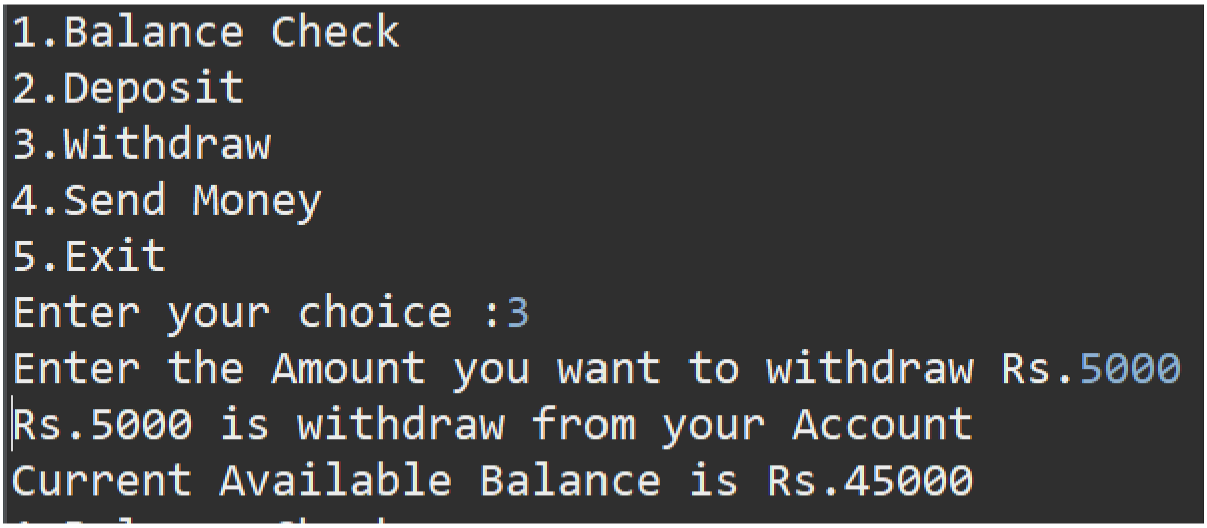
```
Enter your choice :1
Your Current Account Balance is Rs.50000
1.Balance Check
2.Deposit
3.Withdraw
4.Send Money
5.Exit
Enter your choice :
```

4. When user select option 2 for deposit main method will call the deposit method. Deposit method take input amount for deposit and add it to the current amount and print the total amount



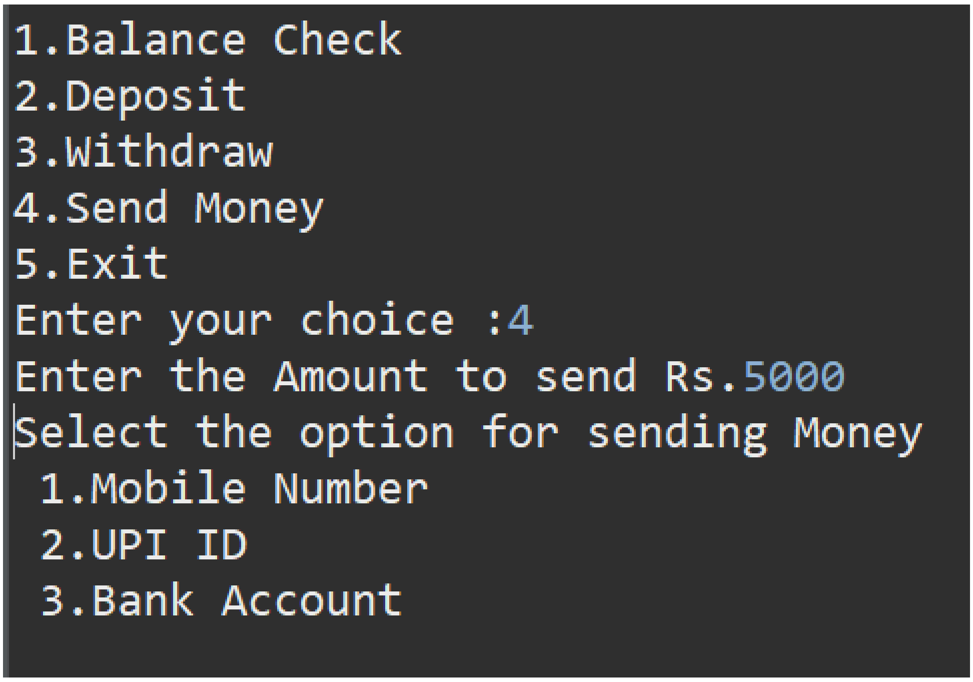
```
1.Balance Check
2.Deposit
3.Withdraw
4.Send Money
5.Exit
Enter your choice :2
Please Enter the amount to Deposit Rs.10000
Rs.10000 is deposited into your Account
Current Available Balance is Rs.60000
```

5. When user select option 3 for withdraw main method will call withdraw method and take input withdraw amount. Withdraw method will check withdraw amount is less than current account balance and deduct it and print the remaining amount



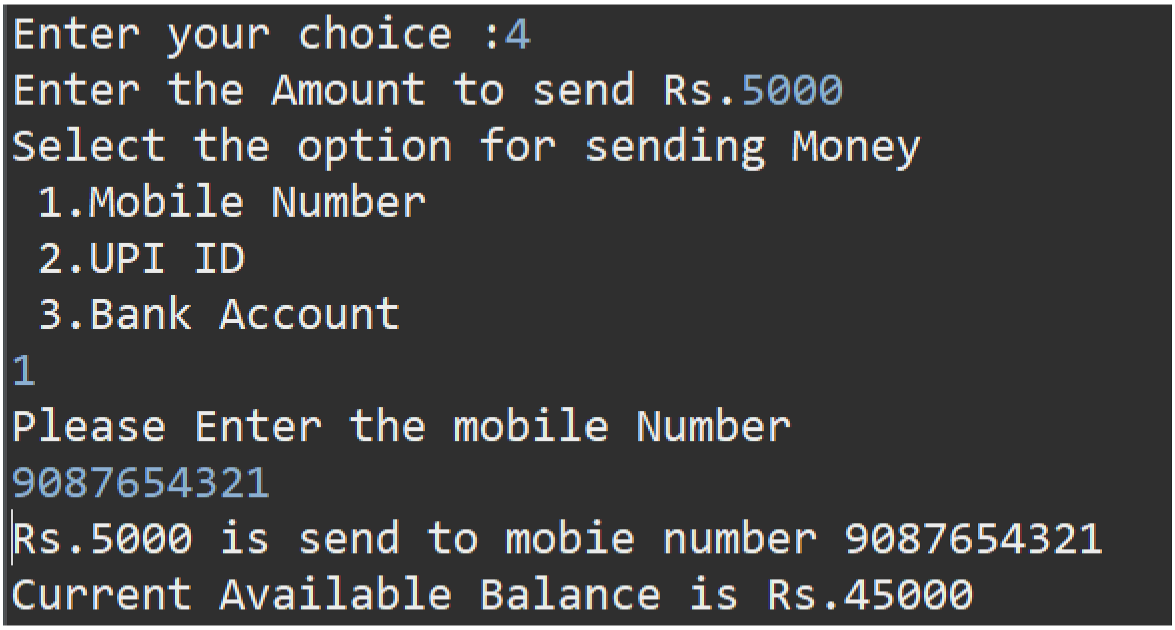
```
1.Balance Check
2.Deposit
3.Withdraw
4.Send Money
5.Exit
Enter your choice :3
Enter the Amount you want to withdraw Rs.5000
Rs.5000 is withdraw from your Account
Current Available Balance is Rs.45000
```

6. When user select option 4 for send money main method will call send method and take input send amount. Send method will check send amount is less than current account balance then console will give user to select the options for sending money



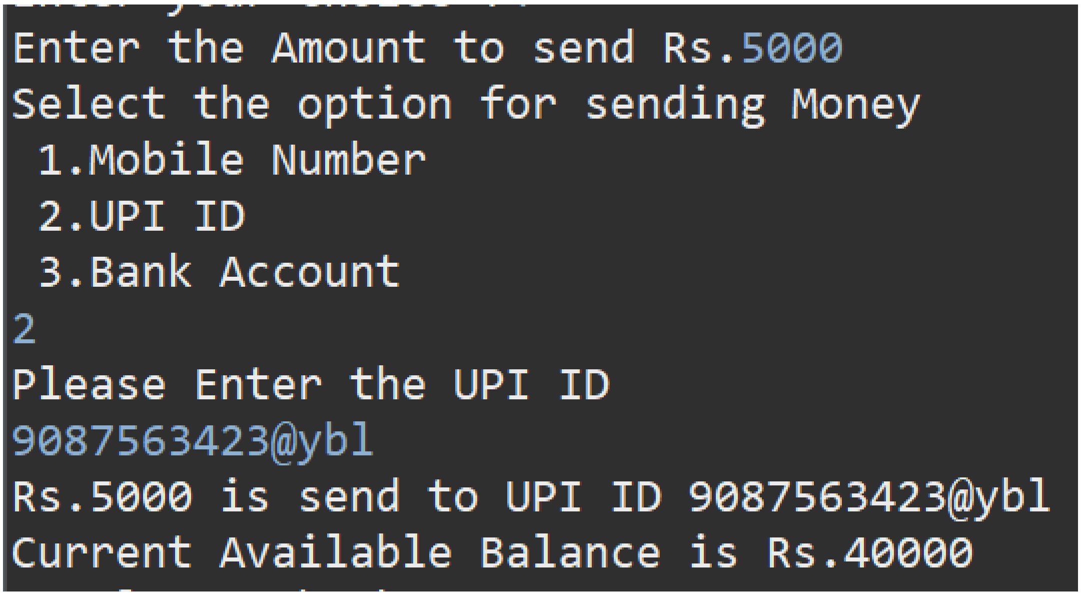
```
1.Balance Check
2.Deposit
3.Withdraw
4.Send Money
5.Exit
Enter your choice :4
Enter the Amount to send Rs.5000
Select the option for sending Money
 1.Mobile Number
 2.UPI ID
 3.Bank Account
```

7. When user select option 1 for mobile number user have to enter mobile number 10 digit otherwise code will give invalid input



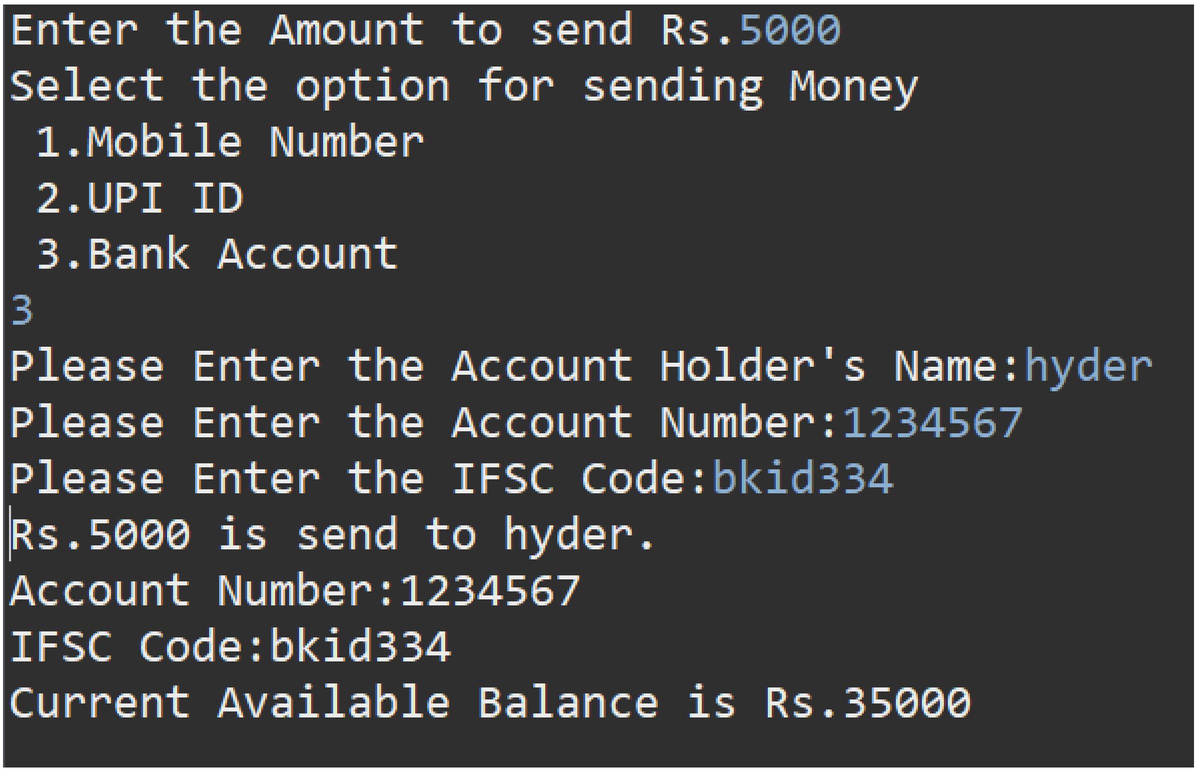
```
Enter your choice :4
Enter the Amount to send Rs.5000
Select the option for sending Money
 1.Mobile Number
 2.UPI ID
 3.Bank Account
1
Please Enter the mobile Number
9087654321
Rs.5000 is send to mobie number 9087654321
Current Available Balance is Rs.45000
```

8. When user select option 2 for UPI Id user have to enter UPI Id with @ in it otherwise code will give invalid input



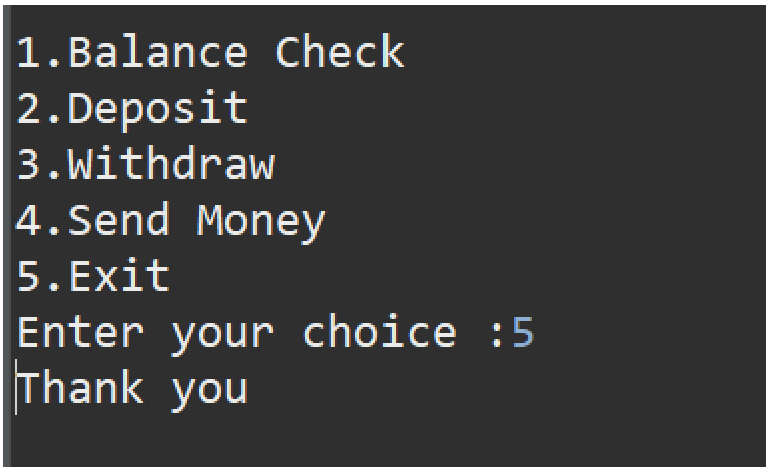
```
Enter the Amount to send Rs.5000
Select the option for sending Money
 1.Mobile Number
 2.UPI ID
 3.Bank Account
2
Please Enter the UPI ID
9087563423@ybl
Rs.5000 is send to UPI ID 9087563423@ybl
Current Available Balance is Rs.40000
```

9. When user select option 3 for bank account user have to enter Bank details otherwise code will give invalid input



```
Enter the Amount to send Rs.5000
Select the option for sending Money
 1.Mobile Number
 2.UPI ID
 3.Bank Account
3
Please Enter the Account Holder's Name:hyder
Please Enter the Account Number:1234567
Please Enter the IFSC Code:bkid334
Rs.5000 is send to hyder.
Account Number:1234567
IFSC Code:bkid334
Current Available Balance is Rs.35000
```

10. When user press option 5 for exit code will be stopped



```
1.Balance Check
2.Deposit
3.Withdraw
4.Send Money
5.Exit
Enter your choice :5
Thank you
```

4 Key performance indicators (KPI)

- Average transaction processing time: This KPI measures the time it takes to complete a transaction. This includes the time it takes for the user to input their transaction details, the processing time on the server, and the time it takes to display the results to the user.
- Number of successful transactions: This KPI measures the number of successful transactions processed by the system. This can provide insights into the system's reliability and efficiency.
- User satisfaction: User satisfaction can be measured through surveys or feedback mechanisms. This KPI provides insights into how well the application meets user needs and expectations.
- Error rate: This KPI measures the percentage of failed transactions due to errors. A high error rate may indicate issues with the application's design or implementation.
- System uptime: This KPI measures the percentage of time that the application is available and functioning properly. System downtime can negatively impact user experience and lead to financial losses.
- Security incidents: This KPI measures the number and severity of security incidents such as unauthorized access, data breaches, and cyber-attacks. Regular monitoring and analysis of this KPI can help identify vulnerabilities and improve the application's security.
- Average response time: This KPI measures the time it takes for the application to respond to user requests. Slow response times can lead to frustrated users and may impact user satisfaction.