High Level Design(HLD)

Console Based Banking Application

Contents

ABSTRACT

The Project Banking Application in java is console based java application.This is somewhat complex Java project consists of five different classes and is a console-based application. When the system starts the user is prompted with a user id and user pin. After entering the details successfully, the ATM functionalities are unlocked and functionalities provide by this application are like Checking Balance, Deposit Money,Withdraw amount and exit..

We Can even use this code for web application by adding some additional features Using JDBC and servlets.We are developing Basic code for Banking Application using eclipse. Additionally, this project is to provide additional features to the user's workspace that are not available in a traditional banking project.

1 .Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to thecurrent project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Re usability
    - Application compatibility
    - Resource utilization
    - Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3 Definitions

Term	Description
IDE	Integrated Development Environment

2 General Description

2.1 Product Perspective

The project main goal is to create an online banking system for banks. All banking work is done manually in the current system. To withdraw or deposit money, the user must go to the bank. Today, it is also hard to find account information for people who have accounts in the banking system.

2.2 Problem statement

This project is to provide additional features to the user's workspace that are not available in a traditional banking project.it provide some basic functionalities like checking balance,withdraw,deposit money and exit console.

2.3 Proposed Solution

The solution proposed here is an console based application using java.without manually going to bank .It provide basic functionality based on the user requirement but this solution is limited to the user console using eclipse.

2.4 Further Improvements

Our project is limited to only one user working only with the help of console input .where here we are not using database to store multiple records of the user and provide service to the multiple user based one there user request .we can even develop this project by using JDBC.and we make this project more advance making it web based application using servlets.

2.5 Technical Requirements

As it is console based basic project we may not require much technology we just require IDE(Integrated Development Environment).So we using eclipse for providing console based application.

2.6 Data Requirements

We don’t require much info externally for the this project we just take username and password as input from the console and perform operation according to the user specification.

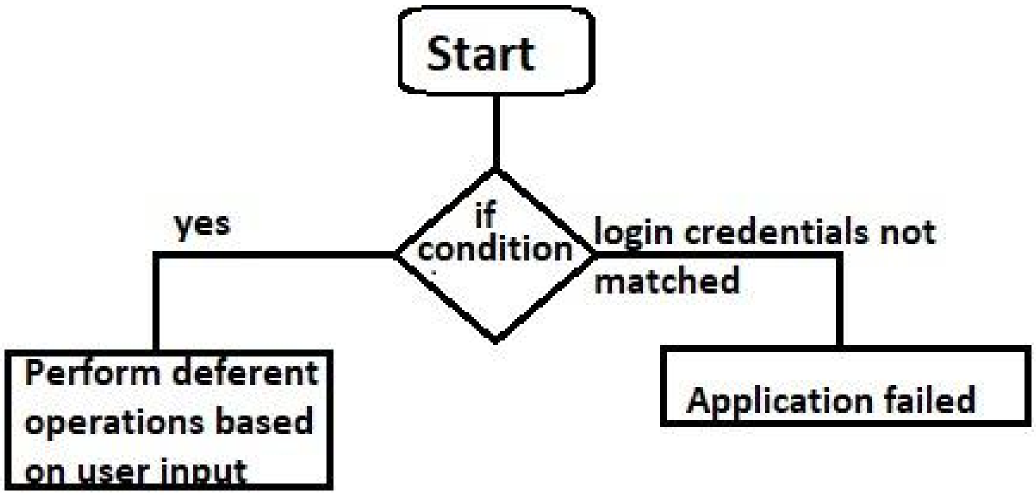
2.7 Tools used

Java programming language and eclipse.

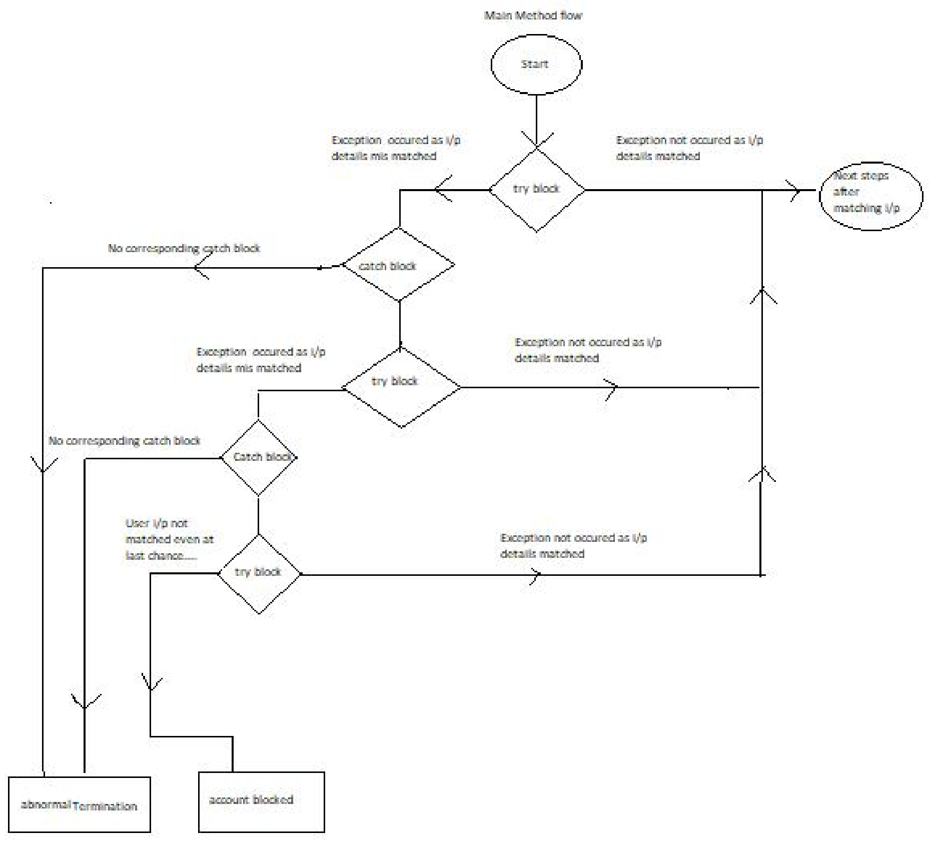
3. Design Details

3.1 Process Flow

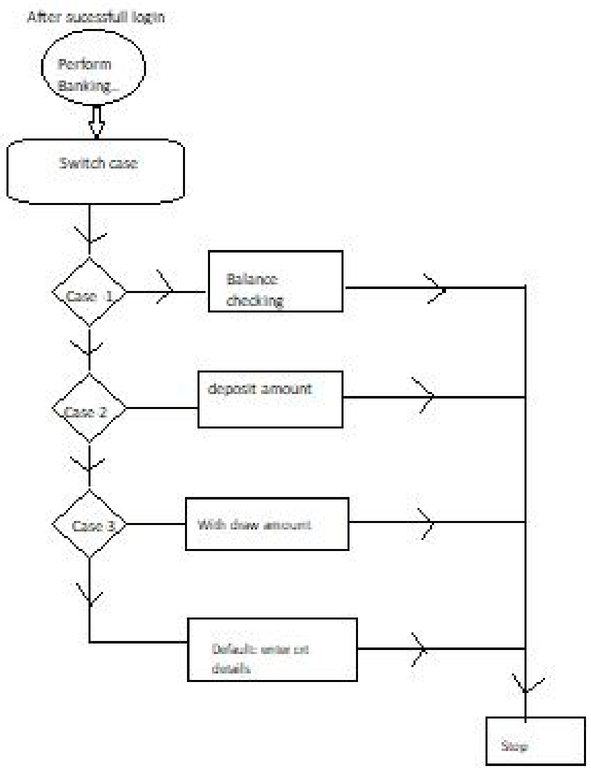
Below is the simple flow diagram for the banking application.


Start
if condition
yes
login credentials not matched
Perform deferent operations based on user input
Application failed

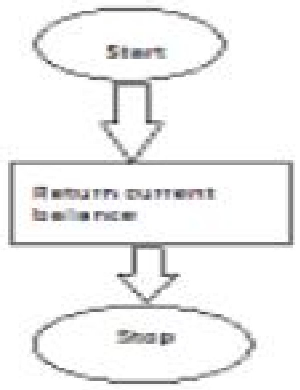
Login Credentials cheacking:


Main Method flow
Start
Exception occured as i/p details mis matched
Exception not occured as i/p details matched
try block
Next steps after matching i/p
No corresponding catch block
catch block
Exception occured as i/p details mis matched
Exception not occured as i/p details matched
try block
No corresponding catch block
Catch block
User i/p not matched even at last chance.....
try block
Exception not occured as i/p details matched
abnormal Termination
account blocked

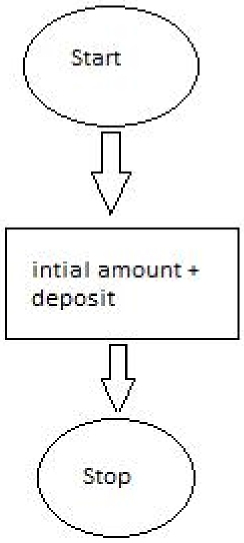
Operation done After success full login :



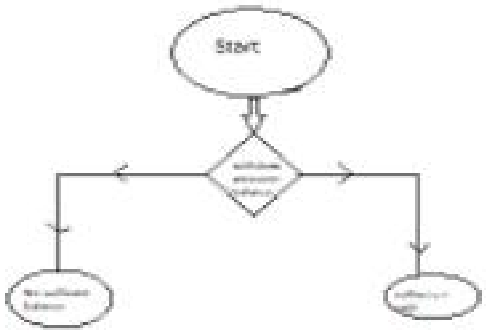
Balance Checking:



Deposit:



With Draw:



4.SYSTEM STUDY AND TECHNOLOGY

4.1 BENEFITS OF ONLINE:

- Time saving.
- Less paper works.
- Cost efficient.
- More comfortable environment.
- Convenience and flexibility

4.2 SOFTWARE REQURIMENTS:

- Java(Programming Language)
- Eclipse(IDE)

5.CODE

```
import java.util.Scanner;

//user defined exception to generate error message if the user
enters wrong credentials
class InvalidCredentialsException extends Exception{
        private static final long serialVersionUID = 1L;

        InvalidCredentialsException(String a){
                super(a);
        }
}

//class to check the balance in the account
class CheckBalance{
        public void currentBalance(int currentBalance) {
                System.out.println("your account balance is
"+currentBalance);
                System.out.println("Thankyou for using our
application.");
        }
}

//class to perform deposit operation on the account
class Deposit{
        public int CurrentBalanceAndDepositedMoney(int
currentBalance, int depositMoney) {
                System.out.println("successfully "+depositMoney+"
was Deposited to your account");
                return currentBalance+depositMoney;
        }
}

//class to perform the withdraw operation on the account
```

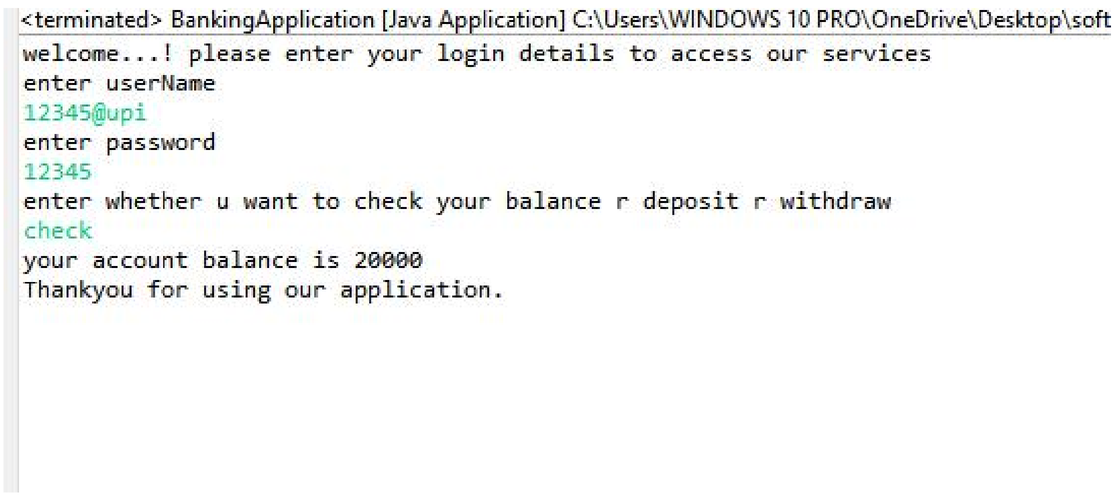
```
class WithDraw{
        public int checkingBalanceToWithdraw(int
currentBalance,int withDrawMoney) {
                if(currentBalance>=withDrawMoney) {
                        System.out.println("successfully
"+withDrawMoney+" was withdrawn from your account");
                        return currentBalance-withDrawMoney;
                }
                else {
                        System.out.println("U dont have enough
balance to withdraw, make sure "
                                        + "your current Balance is greater
than withDrawMoney");
                }
        return currentBalance;
        }
}

public class BankingApplication {

   static Scanner scan=new Scanner(System.in);

   public void performingBankingApplicatio() {
        int currentBalance=20000;
                String option="yes";
                while(option.equalsIgnoreCase("yes")) {
                        System.out.println("enter whether u want to
check your balance r deposit r withdraw or exit ");
                        String value=scan.next();
                switch(value){
                   case "check":
                        new
CheckBalance().currentBalance(currentBalance);
                        break;
                case "deposit":
                    System.out.println("enter how much money u
want to deposit");
                    int depositMoney=scan.nextInt();
                    currentBalance=new
Deposit().CurrentBalanceAndDepositedMoney(currentBalance,
depositMoney);
                    System.out.println("now your balance after
deposit is "+currentBalance);
                    break;

                case "withdraw":
                    System.out.println("enter how much money
you want to withdraw");
                    int withDrawMoney=scan.nextInt();
                    int ModifiedCurrentBalance=new
WithDraw().checkingBalanceToWithdraw(currentBalance,
withDrawMoney);
                    if(ModifiedCurrentBalance<currentBalance) {
                        System.out.println("now your balance
after withdraw is "+ModifiedCurrentBalance);
                    }
                    else {
                          System.out.println("your current balance
is "+ModifiedCurrentBalance);
                    }
                    break;
                case "exit":
                    System.out.println("Successufully loged out
from the application");
                    System.exit(0);
                    break;
                default:
                    System.out.println("enter proper value to
work on your account :(");
```

```
                }
                System.out.println("enter yes if u wnt to use the
application again");
        option=scan.next();
                }if(option.equalsIgnoreCase("yes")!=true) {
                        System.out.println("application terminated...");
                }

   }
   public static void checkingCredentials() throws
InvalidCredentialsException {
        final String Name="12345@upi";
                final String pwd="12345";

                System.out.println("enter userName");
                String userName=scan.next();
                System.out.println("enter password");
                String password=scan.next();
                if(Name.equals(userName) &&
pwd.equals(password)) {
                        new
BankingApplication().performingBankingApplicatio();
                }else {
                        throw new
InvalidCredentialsException("temperarly your account blocked,
you entered wrong credentils too many times ");
                }
   }
        public static void main(String[] args) throws
InterruptedException {
                System.out.println("welcome...! please enter your
login details to access our services ");
                Thread.currentThread().sleep(1000);

                   try {
```

```
                              BankingApplication.checkingCredentials();
                    }catch(Exception e) {
                              try {
                                   System.out.println("credentials
mismatched please enter again");
                                   Thread.currentThread().sleep(1000);

BankingApplication.checkingCredentials();
                              }catch(Exception e1) {
                                   try {
                                           System.out.println("credentials
mismatched please enter again");

Thread.currentThread().sleep(1000);

BankingApplication.checkingCredentials();
                                   }catch(Exception e2) {

System.out.println(e2.getMessage());
                                   }
                              }
                    }

        }
}
```

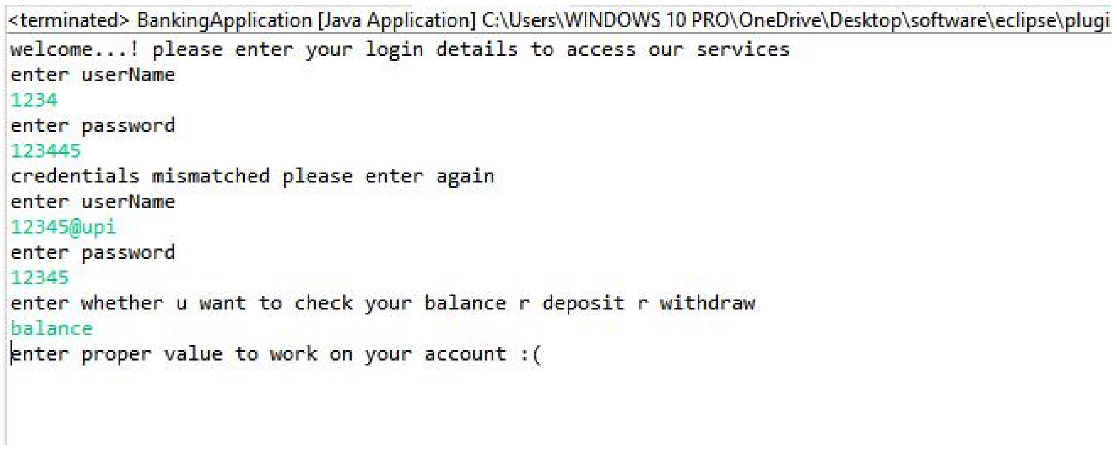
6.OUTPUT

Fig:6.1



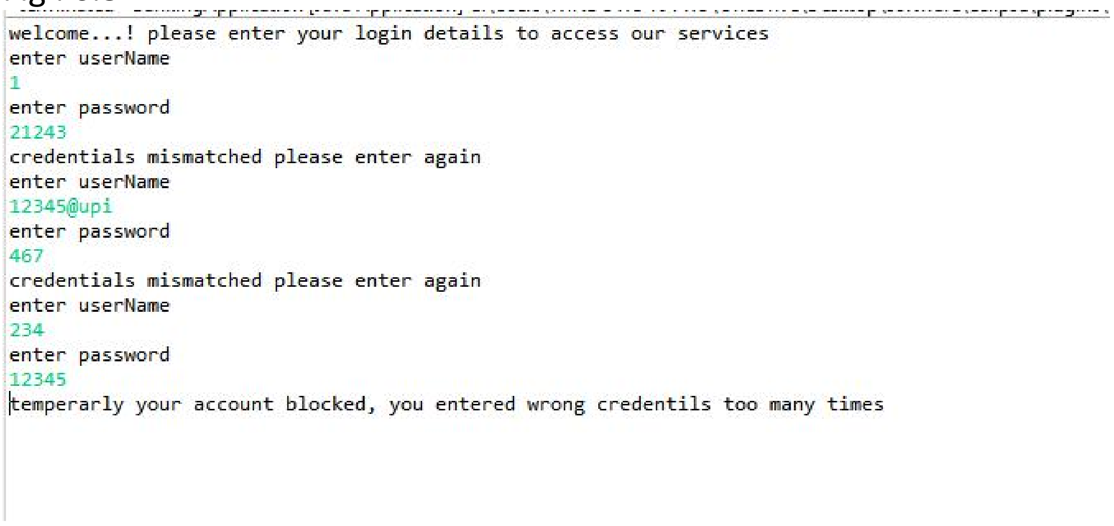
```
<terminated> BankingApplication [Java Application] C:\Users\WINDOWS 10 PRO\OneDrive\Desktop\soft
welcome...! please enter your login details to access our services
enter userName
12345@upi
enter password
12345
enter whether u want to check your balance r deposit r withdraw
check
your account balance is 20000
Thankyou for using our application.
```

Fig : 6.2



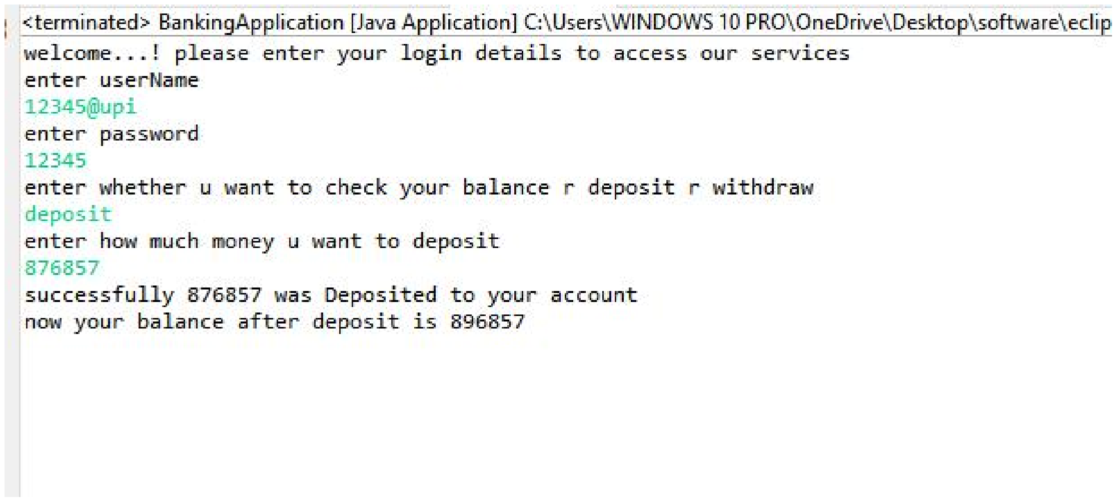
```
<terminated> BankingApplication [Java Application] C:\Users\WINDOWS 10 PRO\OneDrive\Desktop\software\eclipse\plugi
welcome...! please enter your login details to access our services
enter userName
1234
enter password
123445
credentials mismatched please enter again
enter userName
12345@upi
enter password
12345
enter whether u want to check your balance r deposit r withdraw
balance
enter proper value to work on your account :(
```

Fig : 6.3



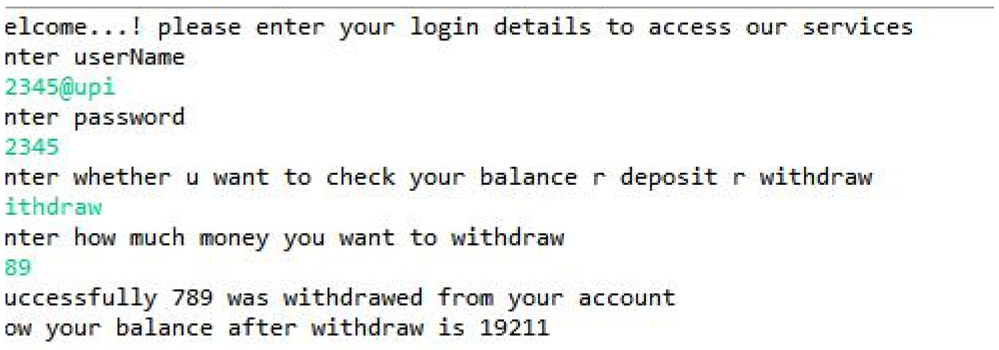
```
welcome...! please enter your login details to access our services
enter userName
1
enter password
21243
credentials mismatched please enter again
enter userName
12345@upi
enter password
467
credentials mismatched please enter again
enter userName
234
enter password
12345
temperarly your account blocked, you entered wrong credentils too many times
```

Fig 6:4



```
<terminated> BankingApplication [Java Application] C:\Users\WINDOWS 10 PRO\OneDrive\Desktop\software\eclip
welcome...! please enter your login details to access our services
enter userName
12345@upi
enter password
12345
enter whether u want to check your balance r deposit r withdraw
deposit
enter how much money u want to deposit
876857
successfully 876857 was Deposited to your account
now your balance after deposit is 896857
```

Fig 6.5


```
elcome...! please enter your login details to access our services
nter userName
2345@upi
nter password
2345
nter whether u want to check your balance r deposit r withdraw
ithdraw
nter how much money you want to withdraw
89
uccessfully 789 was withdrawed from your account
ow your balance after withdraw is 19211
```

7.CONCLUSION

This Basic console based application which help to perform some basic banking applications like Checking Balance,Withdrawing Money and Deposition.