

Banking Management System

**MYBank**



Team Pentagon

Team Members and Roles

Team Representative : Yash Rawat

Team Manager : Sahil Rana

Software Developer : Yash Rawat,

Vishal Verma

Software Tester : Syed Nurja,

Sahil Rana

Yogeshwari Sonavane

**ABSTRACT**

This project is aimed at developing an Online Banking for customer. The system is an online application that can be accessed throughout the organization and outside as well with proper login provided.

The project has been planned to be having the view of distributed architecture, with centralized storage of the database. The application for the storage of the data has been planned. Using the constructs of Oracle 10g and all the user interfaces have been designed using the JAVA. The database connectivity is planned using the “Database” methodology. The standards of security and data protective mechanism have been given a big choice for proper usage. The application takes care of different modules and their associated reports, which are produced as per the applicable strategies and standards that are put forwarded by the administrative staff.

The entire project has been developed keeping in view of the distributed client server computing technology, in mind. The user interfaces are browser specific to give distributed accessibility for the overall system. The internal database has been selected as Oracle 10g.The basic constructs of table spaces, clusters and indexes have been exploited to provide higher consistency and reliability for the data storage. The Oracle 10g was a choice as it provides the constructs of high-level reliability and security. The total front end was dominated using the CONSOLE. At all proper levels high care was taken to check that the system manages the data consistency with proper business rules or validations. The database connectivity was planned using the latest “Database connection” technology provided by Oracle.

**Table of Contents**

1. **INTRODUCTION**
   1. Overview
   2. About the Project
   3. Bank Profile
   4. Functional Components of the Project
   5. Functionality to Enhance the Project
2. **SYSTEM ANALYSIS**
   1. Present System
   2. Proposed System
   3. Benefits of the System
   4. System can be used Extensively
3. **FEASIBILITY REPORT**
   1. Understanding Feasibility
   2. Technical Feasibility
   3. Operational Feasibility
   4. Economic Feasibility
4. **SOFTWARE REQUIREMENT SPECIFICATIONS**
5. **SYSTEM DESIGN**
   1. DFDs
   2. Database Table
6. **CODING**
7. **OUTPUT SCREENS**
8. **SYSTEM TESTING AND IMPLEMENTATION** 
   1. Introduction
   2. Strategic Approach OF Software Testing
   3. Unit Testing
   4. Test
9. **CONCLUSION & FUTURE IMPROVMENT**

**BIBLIOGRAPHY**

**CHAPTER- 1**

**Introduction**

**1.1 Overview**

Internet Banking is all about knowing our customer need and provide them with the right service at the right time through right channel 24\*7 day a week.

Being “electronic”, it not only provides its customers with faster and better facilities, it even reduces the manual overhead of accounts maintenance.

**1.2 ABOUT THE PROJECT**

MYBank is one of the most prestigious BANKs in India. Founded as a Public BANK in 2000 , it is a private institution run by the Pentagon company.

MYBank is affiliated to the Central Board of Bank (CBB), which is the largest educational board in the country. It is recognized by the Department of Education, Govt. of NCT Delhi and the Ministry of HRD, Govt. of India. Over 5000 BANKs in India, with over 80,000 students, are members of the Board.

The BANK is also affiliated to the Indian Public BANKs' Conference (IPSC), and the National Progressive BANKs' Conference (NPSC). The members of these organizations include some of the premier BANKs in the country.

Life at DPSRKP centers on a shared commitment to academic excellence, intellectual growth, art, athletics, high standards of ethical awareness, sportsmanship, and community service. The BANK's traditions and accessibility to a broad curriculum add depth to each student’s life.  
The BANK upholds the founders' commitment to excellence in all fields, with emphasis on its motto Service Before Self.

**1.3 BANK PROFILE:**

The mission of MYBank -“to open doors and open minds” and prepare the ground for the future of the nation.

**1.4 Functional components of the project:**

Following are the functional needs of the software:-

1. Customer must have a valid user ID and password to login to the system.

2. After the valid user logs in, the system shows the present balance in that particular account number.

3. Customer can perform transactions like deposit and withdrawal from his account.

4. Proper help to be provided as and when requested by the customer.

**1.5 More functionality can be added to “enhance the project”:**

1. By adding new modules of different accounts like saving A/C, current A/C etc. to facilitate new customers/users.

2.For new user ,it is must to register first with the correct information asked by the system.

**CHAPTER- 2**

**System Analysis**

**PRESENT SYSTEM**

The developed system is an innovation in the area of banking. In the existing system the no. of staff required for completing the work is more, while the new system requires lesser staffs generally.

The data entry process requires the data on the paper, which is then feed into the application by the operator while doing so; the data entry operator has to look into the paper again & again and thus the chances of in accuracies in the typed contents increases. Also the process includes higher transportation cost, increased handling cost, more time delays, low accuracy, more usage of resources like registers, books, papers, etc.

**PROPOSED SYSTEM**

**“Why an Automated Private Banking System?”**

Almost 50% of today’s information is still paper based.

30% of all office time is spent finding documents.

The average time to manage a single document is 12 minutes,

9 minutes to re-file and 3 minutes to process.

Hence the requirement is to develop a system that minimizes all these overheads included while giving the maximum output for the organization.

The basis for the project is to develop a fully automated banking system that includes depositing of amount,

withdrawal of amount and exporting the outcome back to the clientwhile considering all the tools and facilities than a client may needfor efficient and effective output.

**Benefits of the system**

Quick, authenticated access to accounts via the desktop.

Easily scalable to grow with changing system requirement.

Enterprise wide access to information.

Improved information security, restricting unauthorized access.

Minimize Storage Space

In manual system, much storage space for data files is required so to overcome this problem, on automated well managed

database is developed for saving storage space. This s/w saves space and stores information efficiently. It ends the burden of having large manual filing storage system.

**Banking System can be used extensively**

Withdrawal of amount by the client.

Deposition of amount by the client.

Faster balance enquiry.

**CHAPTER- 3**

**FEASIBILITY REPORT**

**3.1 Understanding Feasibility**

Feasibility study means the analysis of problem to determine if it can be solved effectively. In other words, it is the study of the possibilities of the proposed system it studies the work ability, impact on the organization ability to meet user’s need and efficient use of resources.

**3.2 ECONOMICAL FEASIBILITY:**

The economic analysis checks for the high investment incurred on the system. It evaluates development & implementing charges for the proposed “Banking Management Project”. The S/W used for the development is easily available at minimal cost & the database applied is freely available hence it results in low-cost implementation.

**3.3 TECHNICAL FEASIBILITY:**

The proposed system is technically feasible as it can be developed easily with the help of available technology. The proposed system requires JDBC as an Interface for Programming & back-end as SQL (Oracle 10g) for storing/maintaining database. The database can be easily interconnected using Oracle 10g.

**3.4 BEHAVIOURAL FEASIBILITY:**

Behavioral feasibility deals with the runtime performance of the proposed system must score higher than the present in the behavioral study. It should have end user in mind when the system is designed while designing the programmer should be aware of the conditions user’s Knowledge input, output, calculations etc.

The JDBC contains only a minimum no. of bugs. Care should be also taken to avoid non-working means &t buttons.

**CHAPTER- 4**

**Software Requirement & Specification**

**Software Required:**

The project is implemented in Core Java as it provides the implementation of classes that are used to connect distinct applications, hence the software’s required in the creation and execution of the project is Eclipse .As we know JAVA is a platform independent language so this software runs with JRE environment on any desired platform i.e. Linux ,windows 10, XP, or 2000 or any operating system.

**Hardware Required:**

As the project does not involve any database, its hardware requirements are minimal. Any System with Pentium P2 or above processor, 32MB RAM, 1GB Hard Disk, a LAN Card, and a CDROM is sufficient. Its network-based software so computers connected with any kind of mode (wireless, LAN connected etc.) will suit its requirements. . . . It can also be run on a single machine for its demo use.

Best suited in laboratory where we can run its server on any machine and many clients can use it simultaneously.

**Software Analysis Report**

**About java:**

**Platform Independent:**

The concept of Write-once-run-anywhere (known as the Platform independent) is one of the important key feature of java language that makes java as the most powerful language. Not even a single language is idle to this feature but java is closer to this feature. The programs written on one platform can run on any platform provided the platform must have the JVM.

**Simple:**

There are various features that make the java as a simple language. Programs are easy to write and debug because java does not use the pointers explicitly. It is much harder to write the java programs that can crash the system but we cannot say about the other programming languages. Java provides the bug free system due to the strong memory management. It also has the automatic memory allocation and de-allocation system.

**Object Oriented:**

To be an Object-Oriented language, any language must follow at least the four characteristics.

* Inheritance: It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding the additional features as needed.
* Encapsulation: It is the mechanism of combining the information and providing the abstraction.
* Polymorphism:  As the name suggest one name multiple forms, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.
* Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

As the languages like Objective C, C++ fulfills the above four characteristics yet they are not fully object-oriented languages because they are structured as well as object-oriented languages. But in case of java,  it is a fully Object-Oriented language because object is at the outer most level of data structure in java. No standalone methods, constants, and variables are there in java. Everything in java is object even the primitive data types can also be converted into object by using the wrapper class.

**Robust:**

Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages. Compiler checks the program whether there any error and interpreter check any run time error and makes the system secure from crash. All of the above features make the java language robust.

**Distributed:**

The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access the files from any remote machine on the internet rather than writing codes on their local system.

**Portable:**

The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM. Java also have the standard data size irrespective of operating system or the processor. These features make the java as a portable language.

**Dynamic:**

While executing the java program the user can get the required files dynamically from a local drive or from a computer thousands of miles away from the user just by connecting with the Internet.

**Secure:**

Java does not use memory pointers explicitly. All the programs in java are run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk. Java uses the public key encryption system to allow the java applications to transmit over the internet in the secure encrypted form. The byte code Verifier checks the classes after loading.

**Performance:**

Java uses native code usage, and lightweight process called threads. In the beginning interpretation of byte code resulted the performance slow but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

**Multithreaded:**

Java is also a multithreaded programming language. Multithreading means a single program having different threads executing independently at the same time. Multiple threads execute instructions according to the program code in a process or a program. Multithreading works the similar way as multiple processes run on one computer.    
Multithreading programming is a very interesting concept in Java. In multithreaded programs not even, a single thread disturbs the execution of other thread. Threads are obtained from the pool of available ready to run threads and they run on the system CPUs. This is how Multithreading works in Java which you will soon come to know in details in later chapters.

**Interpreted:**

we all know that Java is an interpreted language as well. With an interpreted language such as Java, programs run directly from the source code.   
The interpreter program reads the source code and translates it on the fly into computations. Thus, Java as an interpreted language depends on an interpreter program.   
The versatility of being **platform independent** makes Java to outshine from other languages. The source code to be written and distributed is platform independent.    
Another advantage of Java as an interpreted language is its error debugging quality. Due to this any error occurring in the program gets traced. This is how it is different to work with Java.

**Architecture Neutral:**

The term architectural neutral seems to be weird, but yes Java is an architectural neutral language as well. The growing popularity of networks makes developers think distributed. In the world of network, it is essential that the applications must be able to migrate easily to different computer systems. Not only to computer systems but to a wide variety of hardware

architecture and operating system architectures as well.  The Java compiler does this by generating byte code instructions, to be easily interpreted on any machine and to be easily translated into native machine code on the fly.The compiler generates an architecture-neutral object file format to enable a Java application to execute anywhere on the network and then the compiled code is executed on many processors, given the presence of the Java runtime system.Hence Java was designed to support applications on network. This feature of Java has thrived the programming language.

**ABOUT : JDK:**

The primary components of the JDK are a selection of programming tools, including:

* java – The [loader](http://en.wikipedia.org/wiki/Loader) for Java applications. This tool is an interpreter and can interpret the class files generated by the [javac](http://en.wikipedia.org/wiki/Javac) compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, is no longer provided with Sun JDK.
* [javac](http://en.wikipedia.org/wiki/Javac) – The [compiler](http://en.wikipedia.org/wiki/Compiler), which converts source code into [Java bytecode](http://en.wikipedia.org/wiki/Java_bytecode)
* jar – The archiver, which packages related class [libraries](http://en.wikipedia.org/wiki/Library_(computer_science)) into a single [JAR file](http://en.wikipedia.org/wiki/Jar_(file_format)). This tool also helps manage JAR files.
* [javadoc](http://en.wikipedia.org/wiki/Javadoc) – The documentation generator, which automatically generates documentation from [source code](http://en.wikipedia.org/wiki/Source_code) comments
* jdb – The [debugger](http://en.wikipedia.org/wiki/Debugger)
* javap – The class file disassembler
* appletviewer – This tool can be used to run and debug Java applets without a web browser.
* javah – The C header and stub generator, used to write native methods
* extcheck – This utility can detect JAR-file conflicts.
* apt – The annotation processing tool
* jhat – (Experimental) Java heap analysis tool
* jstack – (Experimental) This utility prints Java stack traces of Java threads.
* jstat – (Experimental) [Java Virtual Machine](http://en.wikipedia.org/wiki/Java_Virtual_Machine) statistics monitoring tool
* jstatd – (Experimental) jstat daemon
* jinfo – (Experimental) This utility gets configuration information from a running Java process or crash dump.
* jmap – (Experimental) This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump.
* idlj – The IDL-to-Java compiler. This utility generates Java bindings from a given IDL file.
* policy tool – The policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources
* [VisualVM](http://en.wikipedia.org/wiki/VisualVM) – visual tool integrating several command line JDK tools and lightweight performance and memory profiling capabilities

The JDK also comes with a complete [Java Runtime Environment](http://en.wikipedia.org/wiki/Java_Runtime_Environment), usually called a *private* runtime. It consists of a [Java Virtual Machine](http://en.wikipedia.org/wiki/Java_Virtual_Machine) and all of the class libraries that will be present in the production environment, as well as additional libraries only useful to developers, such as the [internationalization](http://en.wikipedia.org/wiki/Internationalization_and_localization) libraries and the [IDL](http://en.wikipedia.org/wiki/Interface_description_language) libraries.

Also included are a wide selection of example programs demonstrating the use of almost all portions of the [Java API](http://en.wikipedia.org/wiki/Java_API).

**About JDBC**

JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually any relational database. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of Java and JDBC lets a programmer write it once and run it anywhere.

* **FUNCTIONS OF JDBC**

Simply put, JDBC makes it possible to do three things:

* Establish a connection with a database
* Send SQL statements
* Process the results.
* **JDBC CONNECTIVITY**

The JDBC provides database-independent connectivity between the J2EE platform and a wide of tabular data sources. JDBC technology allows an Application Component Provider range to:

* Perform connection and authentication to a database server
* Manager transactions
* Move SQL statements to a database engine for preprocessing and execution

Execute stored procedures

**Hardware and Software Requirements**

**Hardware Requirements:**

Processor: Intel Pentium based system

Processor speed:  250 MHz to 833MHz

RAM SPEED:  1GB

**Software Requirements:**

FRONT END: HTML, JAVA SCRIPT

Technologies: CORE JAVA

DATABASE: SQL

Connectivity: JDBC

Operating System: Windows XP/7/8/10

**CHAPTER- 5**

**System Design**

**DATA FLOW**

1. A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated however by two separate arrows since these happen at different type.
2. A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3. A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4. A Data flow to a data store means update (delete or change).
5. A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

User

Valid user

**Banking Management System**

**Database**

Registration Info

Valid Administrator

Login\_ Info

Fig. Level 1 DFD

Registration Info

Verify data

Valid User

Register Info

Update

Login\_Info

Fig . Level 2 DFD process-1

User Data

User Data

CUSTOMER

Personal Details

Account Request

Response

Verified\_ Appl

update

Retrieve

Fig . Level 2 DFD process-2

Valid user

Request for transfer

Valid user

Balance Enquiry

Deposit cash

Other Account

Other Bank

Money Transfer

Account tab

Branch

Fig . Level 2 DFD process-3

Fig . Level 2 DFD process-4

PROCESS 5

Valid user

Loan\_ application

Sanctioned Loan

Response

Update

Loan details

Valid user

Response

Reply

Feedbacks\_ Reply

Reply

Feed Back

Feedback tab

User Feedback

Feedback

Reply

Fig . Level 2 DFD process-5

verify

update

Valid user

Money\_ Transfer

account tab

account tab

Status Info

Fig . Level 3 DFD

**5.5 DATABASE TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Data type** | **Nullable** | **Primary key** |
| FIRST\_NAME | Varchar (20) | Yes | No |
| LAST\_NAME | Varchar (20) | Yes | No |
| USERNAME | Varchar (20) | Yes | No |
| PASSWORD | Varchar (20) | Yes | No |
| MOBILE\_NUMBER | Varchar (20) | Yes | No |
| E-MAIL | Varchar (20) | Yes | No |
| ADDRESS | Varchar (20) | Yes | No |
| PANCARD\_NUMBER | Varchar (20) | Yes | No |
| ADHAARCARD\_NUMBER | Varchar (20) | Yes | No |
| ACCOUNT | Varchar (20) | Yes | No |
| GENDER | Varchar (20) | Yes | No |
| OCCUPATION | Varchar (20) | Yes | No |

**CHAPTER- 6**

**CODING**

**import java.text.NumberFormat;**

**import java.util.Scanner;**

**import Dashboard.Dashboard;**

**import Loan.Loan;**

**public class Main {**

**public static void main(String[] args) throws Exception {**

**Registration obj = new Registration();**

**obj.registration("vishal", "verma", "vishal123", "vishal@123", "1234567890", "vishal@gmail.com","BoulevardStreet,Bhopal","CYGAZ8549M","698585714489","Savings","Male","BusinessMan");**

**String adhaar\_card = "";**

**String password;**

**String mobile\_No="";**

**String First\_name;**

**String last\_name;**

**String userName;**

**String email;**

**Scanner sc =new Scanner(System.in);**

**System.out.println(" ✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥ Welcome ✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥✥ ");**

**System.out.println();**

**System.out.printf(" ➨ 1. Registration \n \n ➨ 2. Login ");**

**int ch =sc.nextInt();**

**switch(ch) {**

**case 1:**

**System.out.println("Enter your First Name ✍");**

**First\_name=sc.next();**

**System.out.println("Enter your Last Name ✍");**

**last\_name=sc.next();**

**System.out.println("Enter your userName ✍");**

**userName=sc.next();**

**System.out.println("Enter your password ✍");**

**password=sc.next();**

**System.out.println("Enter your Re\_enter password ✍");**

**String re\_pass =sc.next();**

**System.out.println("Enter your Mobile Number ✍");**

**mobile\_No =sc.next();**

**System.out.println("Enter your Email ✉");**

**email=sc.next();**

**System.out.println("Enter your Address ✍");**

**String address=sc.next();**

**System.out.println("Enter your Pancard ✍");**

**String pancard=sc.next();**

**System.out.println("Enter your Adhaar Card ✍");**

**adhaar\_card=sc.next();**

**System.out.println("Select your Account ✍");**

**System.out.printf(" ➡ 1. Saving \n ➡ 2. Current ");**

**int acc\_ch =sc.nextInt();**

**switch(acc\_ch) {**

**case 1:**

**String acc\_type ="Saving";**

**break;**

**case 2:**

**acc\_type ="Current";**

**break;**

**default:**

**System.out.println(" 😭😭😭😭😭 Select valid options 😭😭😭😭😭😭 ");**

**break;**

**}**

**System.out.println("Select your Gender ✍");**

**System.out.printf(" ➡ 1. Male \n ➡ 2. Female \n ➡ 3. Transgender");**

**int gen\_ch=sc.nextInt();**

**switch(gen\_ch) {**

**case 1:**

**String gender = "Male";**

**break;**

**case 2:**

**gender = "Female";**

**break;**

**case 3:**

**gender = "Transgender";**

**break;**

**default:**

**System.out.println(" 😭😭😭😭😭 Select valid options 😭😭😭😭😭😭 ");**

**break;**

**}**

**System.out.println("Enter your Occupation ✍");**

**String occupation =sc.next();**

**System.out.println("Code ends");**

**break;**

**//login function**

**case 2:**

**System.out.println();**

**System.out.println("Enter your UserName ✍");**

**userName=sc.next();**

**System.out.println("Enter your password ✍");**

**password=sc.next();**

**boolean v = obj.login(userName,password);**

**//System.out.println(v);**

**if(v) {**

**//dashboard**

**Dashboard dash = new Dashboard ( );**

**dash.home(userName);**

**System.out.println();**

**boolean temp =false;**

**do {**

**System.out.print(" ➡ 1. Check Balance \n ➡ 2. Get Account Number \n ➡ 3. Transfer Money \n ➡ 4. Bills & Recharges \n ➡ 5. Loan \n ➡ 6. Offers \n ➡7.Support");**

**System.out.println();**

**int ch1 =sc.nextInt();**

**int exit;**

**switch(ch1) {**

**case 1:**

**dash.check\_blance();**

**System.out.println();**

**System.out.println(" ➡ 1. Exit");**

**exit= sc.nextInt();**

**if(exit==1) {**

**temp =true;**

**}**

**break;**

**case 2:**

**dash.gen\_acc();**

**System.out.println();**

**System.out.println(" ➡ 1. Exit");**

**exit= sc.nextInt();**

**if(exit==1) {**

**temp =true;**

**}**

**break;**

**case 3:**

**System.out.println();**

**System.out.println(" ➡ 1. Exit");**

**exit= sc.nextInt();**

**if(exit==1) {**

**temp =true;**

**}**

**break;**

**case 4:**

**System.out.println();**

**System.out.println(" ➡ 1. Exit");**

**exit= sc.nextInt();**

**if(exit==1) {**

**temp =true;**

**}**

**break;**

**case 5:**

**System.out.printf(" ➡ 1. Car Loan \n ➡ 2. Home loan \n ➡ 3. Education Loan \n ➡ 4. Gold Loan ");**

**int ch3=sc.nextInt();**

**Loan obj3 = new Loan();**

**System.out.println();**

**switch(ch3) {**

**case 1:**

**System.out.println("Enter your Adhaar Card ✍");**

**adhaar\_card=sc.next();**

**System.out.println("Enter your Pancard ✍");**

**pancard=sc.next();**

**boolean a= obj3.car\_loan( adhaar\_card,pancard);**

**if(a=true) {**

**System.out.println();**

**System.out.printf("➡ 1. Proceed \n ➡ 2. Cancel");**

**int ch4 =sc.nextInt();**

**switch(ch4) {**

**case 1:**

**break;**

**case 2:**

**temp =true;**

**break;**

**}**

**}**

**break;**

**case 2:**

**System.out.println();**

**System.out.println("Enter your Adhaar Card ✍");**

**adhaar\_card=sc.next();**

**System.out.println("Enter your Pancard ✍");**

**pancard=sc.next();**

**obj3.home\_loan( adhaar\_card,pancard);**

**a= obj3.home\_loan( adhaar\_card,pancard);**

**if(a=true) {**

**System.out.println();**

**System.out.printf("➡ 1. Proceed \n ➡ 2. Cancel");**

**int ch4 =sc.nextInt();**

**switch(ch4) {**

**case 1:**

**break;**

**case 2:**

**temp =true;**

**break;**

**}**

**}**

**break;**

**case 3:**

**System.out.println();**

**System.out.println("Enter your Adhaar Card ✍");**

**adhaar\_card=sc.next();**

**System.out.println("Enter your Pancard ✍");**

**pancard=sc.next();**

**obj3.education\_loan(adhaar\_card,pancard );**

**a= obj3.car\_loan( adhaar\_card,pancard);**

**if(a=true) {**

**System.out.printf("➡ 1. Proceed \n ➡ 2. Cancel");**

**int ch4 =sc.nextInt();**

**switch(ch4) {**

**case 1:**

**break;**

**case 2:**

**temp =true;**

**break;**

**}**

**}**

**break;**

**case 4:**

**System.out.println();**

**System.out.println("Enter your Adhaar Card ✍");**

**adhaar\_card=sc.next();**

**System.out.println("Enter your Pancard ✍");**

**pancard=sc.next();**

**obj3.Gold\_loan(adhaar\_card, pancard );**

**a= obj3.car\_loan( adhaar\_card,pancard);**

**if(a=true) {**

**System.out.printf("➡ 1. Proceed \n ➡ 2. Cancel");**

**int ch4 =sc.nextInt();**

**switch(ch4) {**

**case 1:**

**break;**

**case 2:**

**temp =true;**

**break;**

**}**

**}**

**break;**

**default:**

**break;**

**}**

**break;**

**case 6:**

**break;**

**case 7:**

**break;**

**default:**

**System.out.println(" 😭😭😭😭😭 Select valid options 😭😭😭😭😭😭 ");**

**break;**

**}**

**}while(temp);**

**}**

**else {**

**System.out.println("Forget Your Password ✍");**

**System.out.printf(" ➡ 1. Yes ✔ \n ➡ 2. No ✘");**

**int f\_ch =sc.nextInt();**

**switch(f\_ch) {**

**case 1:**

**break;**

**case 2:**

**break;**

**default:**

**System.out.println(" 😭😭😭😭😭 Select valid options 😭😭😭😭😭😭 ");**

**break;**

**} }**

**break;**

**default:**

**System.out.println(" 😭😭😭😭😭 Select valid options 😭😭😭😭😭😭 ");**

**break;**

**}**

**}**

**}**

**import java.sql.\*;**

**public class Support {**

**public void userDetails(String firstName,String lastName,String email,String message) throws ClassNotFoundException, SQLException {**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","admin","123456");**

**String sql = "insert into support values(?,?,?,?)";**

**PreparedStatement st = con.prepareStatement(sql);**

**st.setString(1, firstName);**

**st.setString(2, lastName);**

**st.setString(3, email);**

**st.setString(4, message);**

**st.executeQuery();**

**st.close();**

**con.close();**

**}**

**}**

**//////////////////////registrasion**

**import java.sql.\*;**

**public class Registration {**

**public void registration(String firstName,String lastName,String username,String password,String mobileNumber,String email,String address,String panNumber, String adhaarNumber,String account,String gender,String occupation) throws Exception {**

**try {**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");**

**String sql = "insert into registration values(?,?,?,?,?,?,?,?,?,?,?,?)";**

**PreparedStatement st = con.prepareStatement(sql);**

**st.setString(1, firstName);**

**st.setString(2, lastName);**

**st.setString(3, username);**

**st.setString(4, password);**

**st.setString(5, mobileNumber);**

**st.setString(6, email);**

**st.setString(7, address);**

**st.setString(8,panNumber);**

**st.setString(9,adhaarNumber);**

**st.setString(10, account);**

**st.setString(11, gender);**

**st.setString(12, occupation);**

**st.executeQuery();**

**st.close();**

**con.close();**

**}catch(Exception e) {**

**e.printStackTrace();**

**}**

**}**

**public boolean login(String username,String password) throws Exception {**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con =**

**DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");**

**//System.out.println(con);**

**String sq = "select \* from registration where username='"+username+"'";**

**Statement st = con.createStatement();**

**ResultSet rs = st.executeQuery(sq);**

**String pass="";**

**while(rs.next()) {**

**rs.getString(1);**

**rs.getString(2);**

**rs.getString(3);**

**pass = rs.getString(4);**

**}**

**if(pass.equals(password))**

**return true;**

**else**

**return false;**

**}**

**}**

**/////////////////////////////////////// Loan**

**mport java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.sql.Statement;**

**public class Loan {**

**private String username;**

**private String adhaar\_card\_number;**

**private String mobile\_number ;**

**private String pan\_card\_number;**

**String adhaar\_card\_number1;**

**String mobile\_number1;**

**public boolean car\_loan(String adhaar\_card\_number,String pan\_card\_number) throws ClassNotFoundException, SQLException {**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");**

**String sq = "select \* from registration where username='"+username+"'";**

**Statement st = con.createStatement();**

**ResultSet rs = st.executeQuery(sq);**

**while(rs.next()) {**

**adhaar\_card\_number1=rs.getString(9);**

**mobile\_number1=rs.getString(5);**

**}**

**if(adhaar\_card\_number.equals(adhaar\_card\_number1)&&mobile\_number.equals(mobile\_number1)){**

**return true;**

**}**

**else {**

**return false;**

**}**

**}**

**public boolean home\_loan(String adhaar\_card\_number,String pan\_card\_number) throws ClassNotFoundException, SQLException {**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");**

**String sq = "select \* from registration where username='"+username+"'";**

**Statement st = con.createStatement();**

**ResultSet rs = st.executeQuery(sq);**

**while(rs.next()) {**

**adhaar\_card\_number1=rs.getString(9);**

**mobile\_number1=rs.getString(5);**

**}**

**if(adhaar\_card\_number.equals(adhaar\_card\_number1)&&mobile\_number.equals( mobile\_number1)){**

**return true;**

**}**

**else {**

**return false;**

**}**

**}**

**public boolean Gold\_loan(String adhaar\_card\_number,String pan\_card\_number) throws ClassNotFoundException, SQLException {**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");**

**String sq = "select \* from registration where username='"+username+"'";**

**Statement st = con.createStatement();**

**ResultSet rs = st.executeQuery(sq);**

**while(rs.next()) {**

**adhaar\_card\_number1=rs.getString(9);**

**mobile\_number1=rs.getString(5);**

**}**

**if(adhaar\_card\_number.equals(adhaar\_card\_number1)&&mobile\_number.equals(mobile\_number1)){**

**return true;**

**}**

**else {**

**return false;**

**}**

**}**

**public boolean education\_loan(String adhaar\_card\_number,String pan\_card\_number) throws ClassNotFoundException, SQLException {**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");**

**String sq = "select \* from registration where username='"+username+"'";**

**Statement st = con.createStatement();**

**ResultSet rs = st.executeQuery(sq);**

**while(rs.next()) {**

**adhaar\_card\_number1=rs.getString(9);**

**mobile\_number1=rs.getString(5);**

**}**

**if(adhaar\_card\_number.equals(adhaar\_card\_number1)&&mobile\_number.equals(mobile\_number1)){**

**return true;**

**}**

**else {**

**return false;**

**}**

**}**

**}**

**////////////////////// Dashboard**

**package Dashboard;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.sql.Statement;**

**import java.util.Random;**

**import java.util.Scanner;**

**public class Dashboard {**

**Scanner sc =new Scanner(System.in);**

**private String pass;**

**private String Account\_number;**

**private String adhaar\_card\_number;**

**private String mobile\_number ;**

**String username1;**

**public void home(String username) throws ClassNotFoundException, SQLException {**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");**

**String sq = "select \* from registration where username='"+username+"'";**

**Statement st = con.createStatement();**

**ResultSet rs = st.executeQuery(sq);**

**String pass="";**

**while(rs.next()) {**

**username1 = rs.getString(1);**

**adhaar\_card\_number=rs.getString(9);**

**mobile\_number=rs.getString(5);**

**}**

**System.out.println(" ✥✥✥✥✥✥✥ Hello "+this.username1+" ✥✥✥✥✥✥✥✥✥✥");**

**}**

**public void check\_blance() {**

**System.out.println("Enter your Account Number ✍");**

**Account\_number=sc.next();**

**System.out.println("Enter your Password ✍");**

**pass=sc.next();**

**}**

**public void gen\_acc() throws ClassNotFoundException, SQLException {**

**String acc\_number=adhaar\_card\_number.substring(0,7) +mobile\_number .substring(0,7);**

**System.out.println();**

**System.out.println(" ➡ Your Account number is "+acc\_number);**

**}**

**/\***

**\* public String getUser\_name() { return user\_name; } public void**

**\* setUser\_name(String user\_name) { this.user\_name = user\_name; } public String**

**\* getPass() { return pass; } public void setPass(String pass) { this.pass =**

**\* pass; } public String getAccount\_number() { return Account\_number; } public**

**\* void setAccount\_number(String account\_number) { this.Account\_number =**

**\* account\_number; }**

**\***

**\* public String getAdhaar() { return Adhaar; } public void setAdhaar(String**

**\* adhaar) { this.Adhaar = adhaar; } public String getMobile() { return mobile;**

**\* } public void setMobile(String mobile) { this.mobile = mobile; } public**

**\* Dashbord(String user\_name, String pass, String mobile, String adhaar) {**

**\* super(); this.user\_name = user\_name; this.pass = pass; this.mobile = mobile;**

**\***

**\* this.Adhaar = adhaar; }**

**\*/**

**}**

**import java.sql.\*;**

**public class Support {**

**public void userDetails(String firstName,String lastName,String email,String message) throws ClassNotFoundException, SQLException {**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","admin","123456");**

**String sql = "insert into support values(?,?,?,?)";**

**PreparedStatement st = con.prepareStatement(sql);**

**st.setString(1, firstName);**

**st.setString(2, lastName);**

**st.setString(3, email);**

**st.setString(4, message);**

**st.executeQuery();**

**st.close();**

**con.close();**

**}**

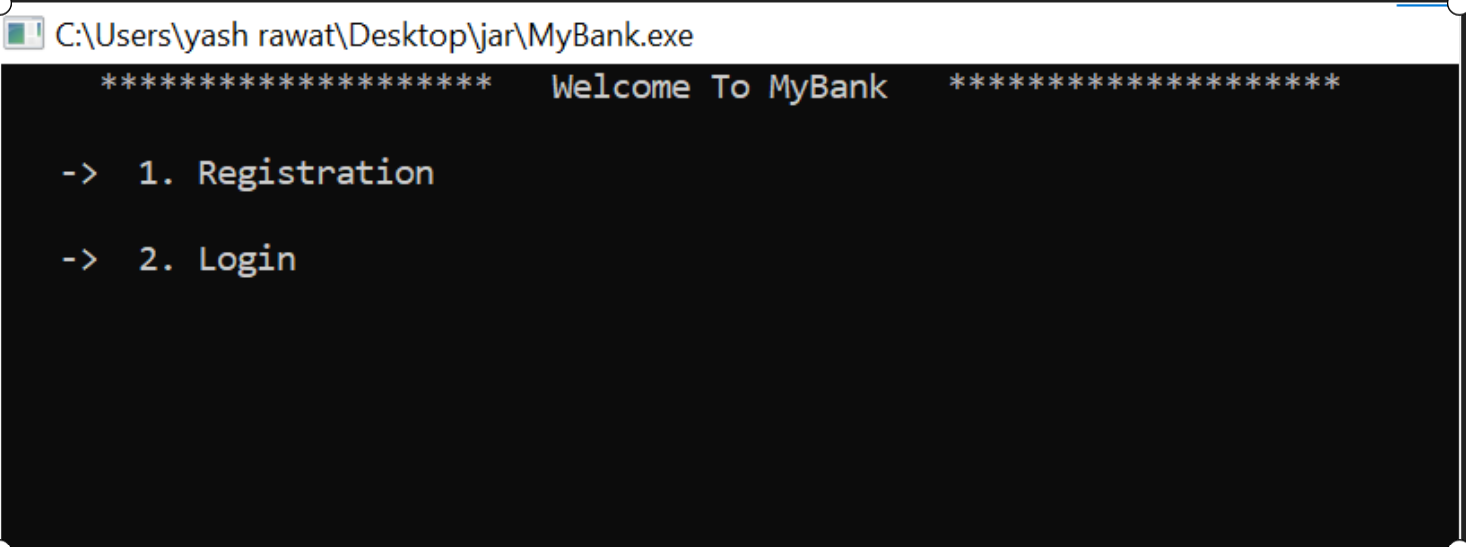
**}**

**CHAPTER- 7**

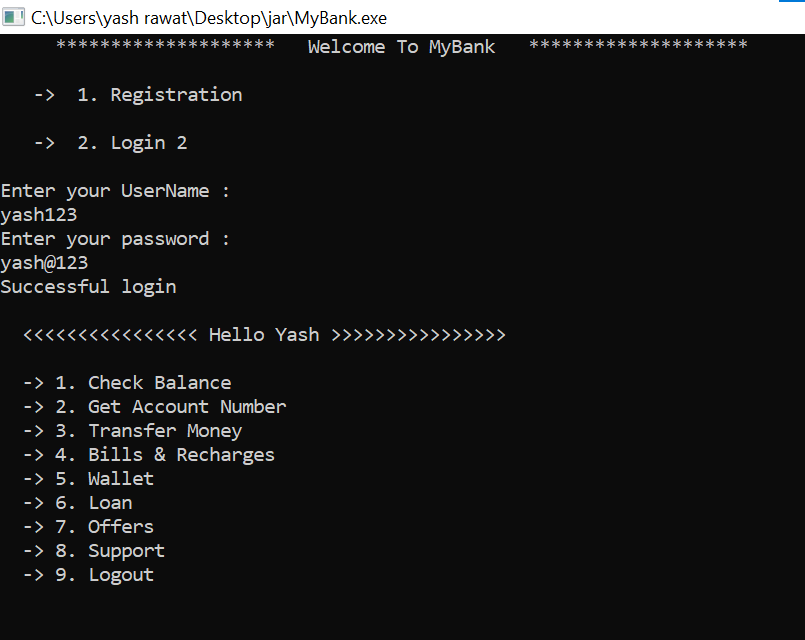
**OUTPUT SCREENS**

**Steps:**

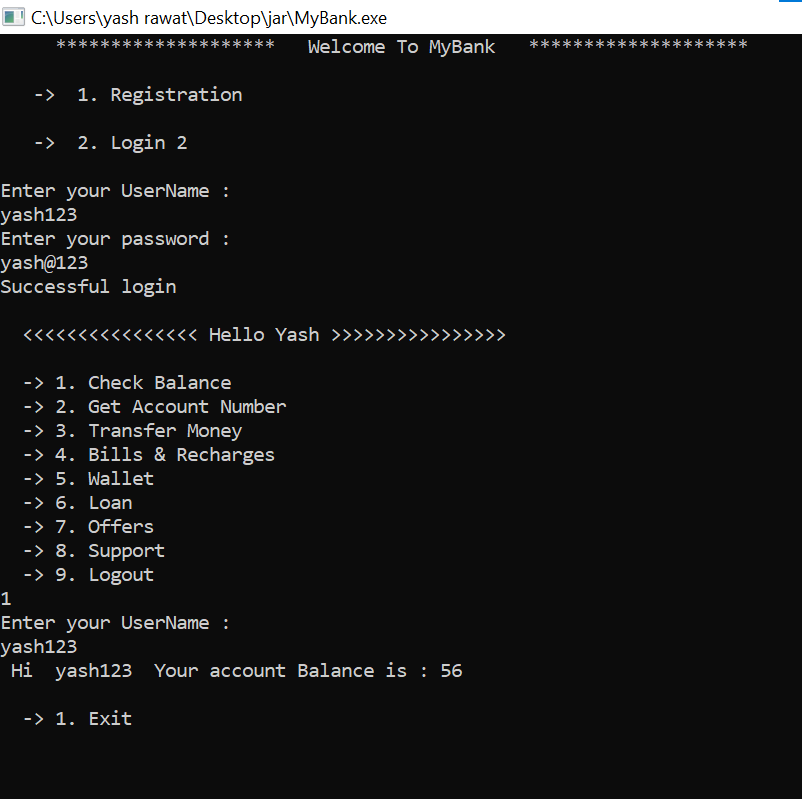
1. **Create new account by clicking on Registration.**



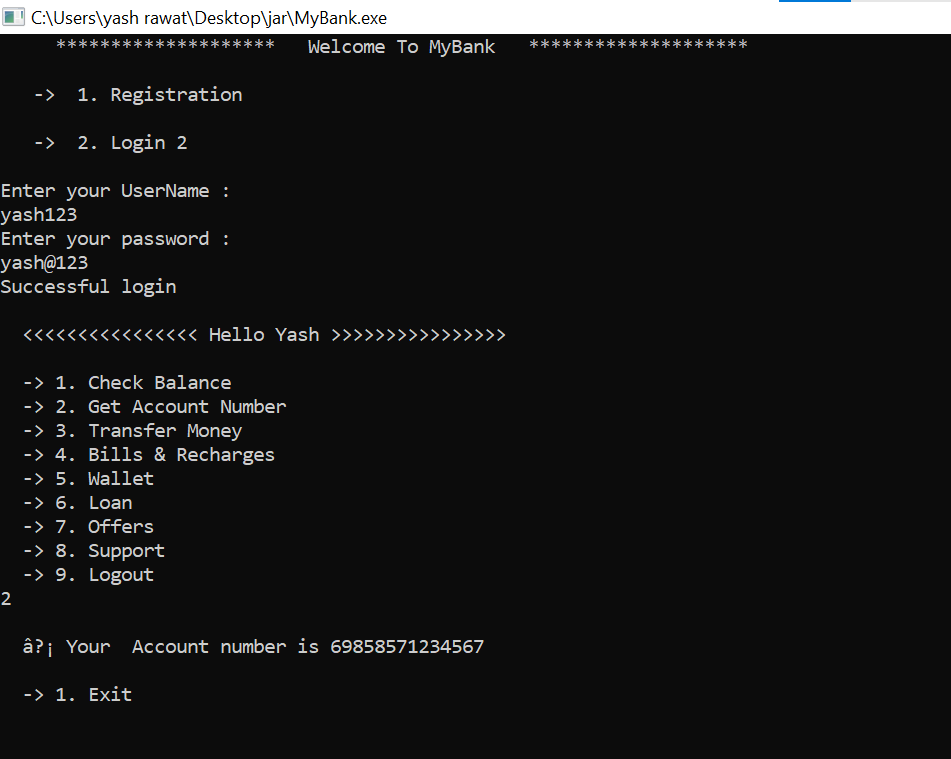
**2. Successfully login :**

****

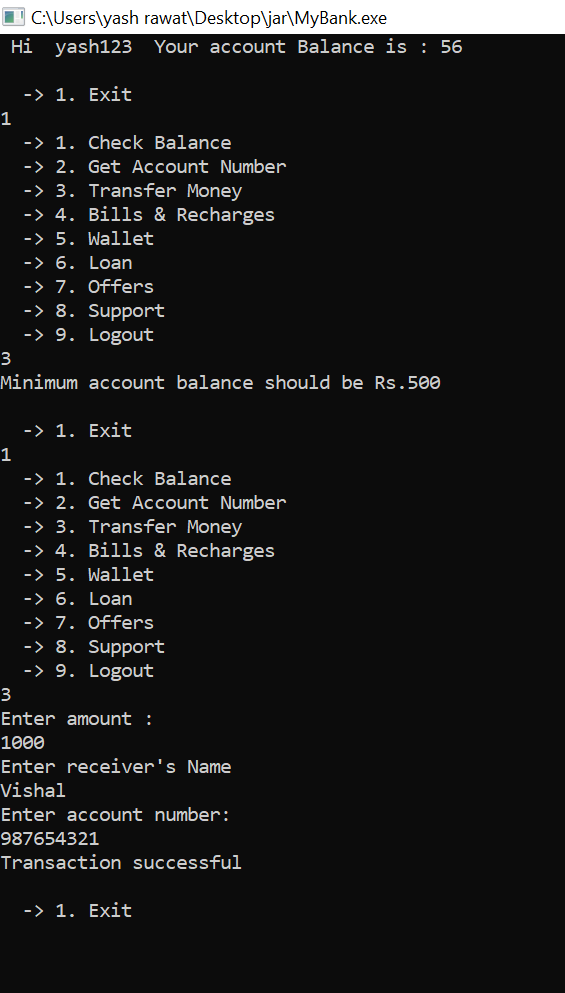
**3. Check Balance**

****

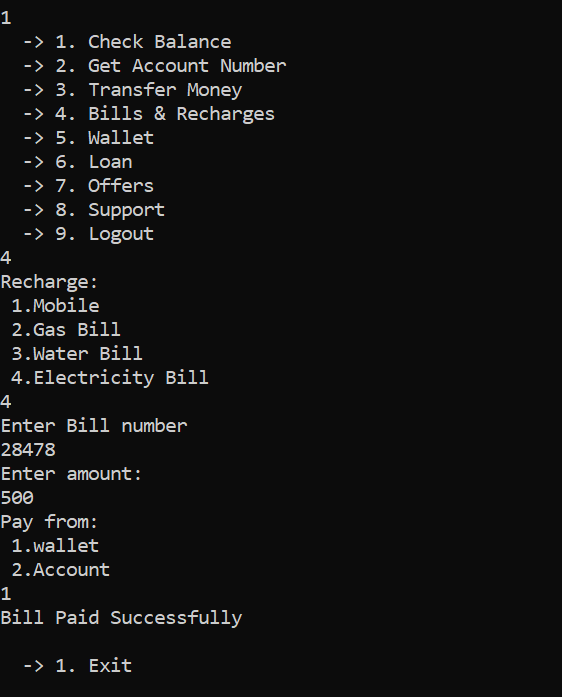
**4. Get Account Number**

****

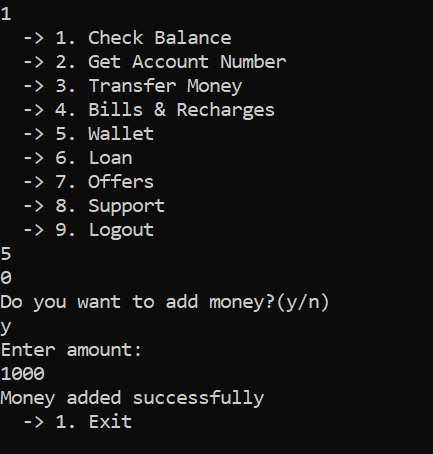
**5. Transfer Money**

****

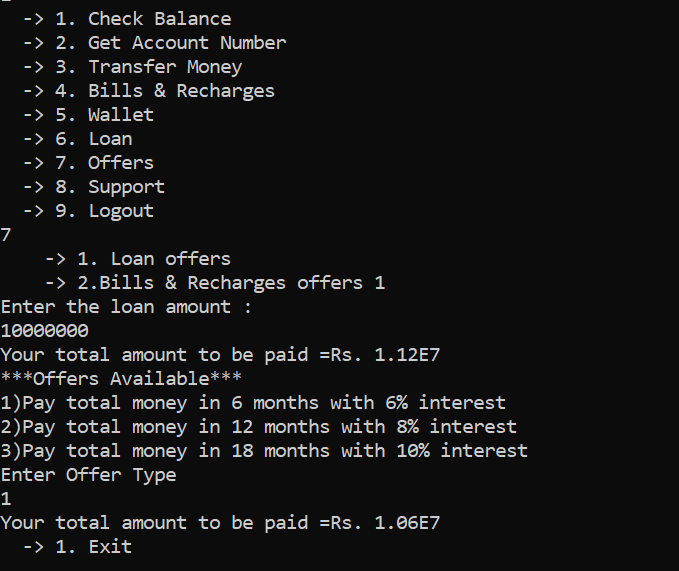
**6. Bills and Recharges**

****

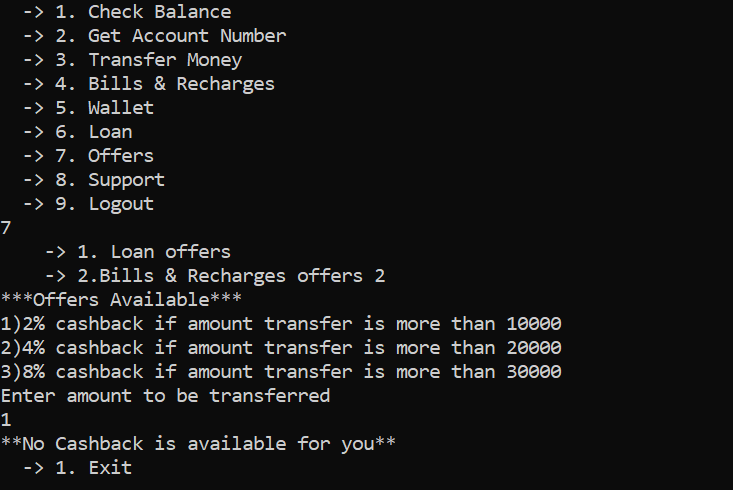
**7. Wallets**

****

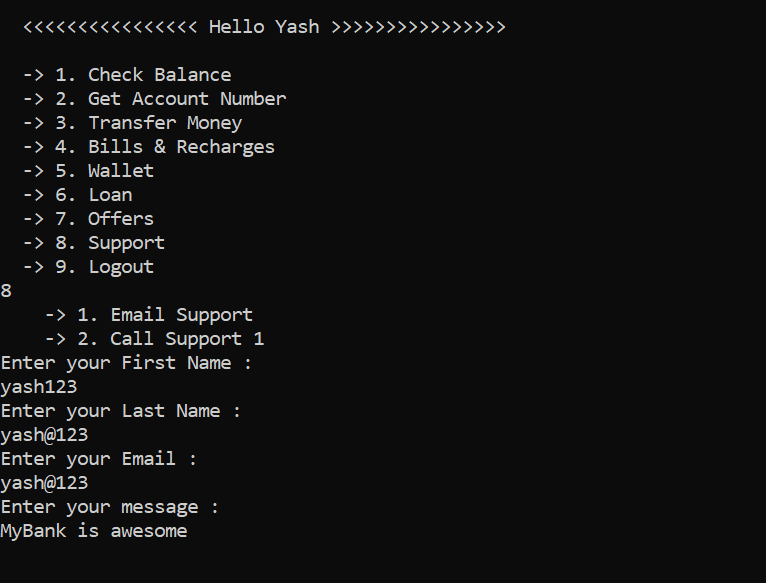
**8. Loan**

****

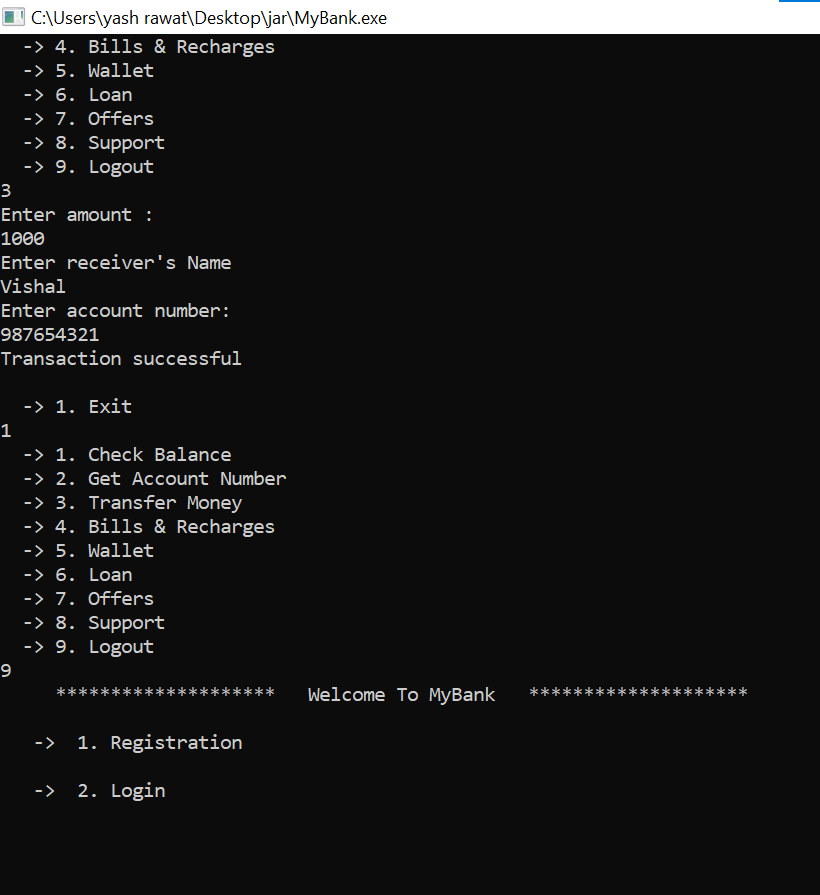
**9. Offers**

****

**10. Support**

****

**11. Logout**

****

**CHAPTER- 8**

**SYSTEM TESTING AND IMPLIMENTATION**

**8.1. INTRODUCTION**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

**8.2. STRATEGIC APPROACH TO SOFTWARE TESTING**

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

UNIT TESTING

MODULE TESTING

SUB-SYSTEM TESING

SYSTEM TESTING

ACCEPTANCE TESTING

Component Testing

Integration Testing

User Testing

**8.3. Unit Testing**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

**1. WHITE BOX TESTING**

This type of testing ensures that

* All independent paths have been exercised at least once
* All logical decisions have been exercised on their true and false sides
* All loops are executed at their boundaries and within their operational bounds
* All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

**2. BASIC PATH TESTING**

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

V(G)=E-N+2 or

V(G)=P+1 or

V(G)=Number Of Regions

Where V(G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

**3. CONDITIONAL TESTING**

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

**4. DATA FLOW TESTING**

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

**5. LOOP TESTING**

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

* All the loops were tested at their limits, just above them and just below them.
* All the loops were skipped at least once.
* For nested loops test the inner most loop first and then work outwards.
* For concatenated loops the values of dependent loops were set with the help of connected loop.
* Unstructured loops were resolved into nested loops or concatenated loops and tested as above.

Each unit has been separately tested by the development team itself and all the input have been validated.

**CHAPTER- 10**

**CONCLUSION & SCOPE FOR FUTRURE DEVLOPMENT**

**CONCLUSION**

This project developed, incorporated all the activities involved in the browsing centre.

It provides all necessary information to the management as well as the customer with the use of this system; the user can simply sit in front of the system and monitor all the activities without any physical movement of the file. Management can service the customers request best in time.

The system provides quickly and valuable information. These modules have been integrated for effective use of the management for future forecasting and for the current need.

**SCOPE FOR FURTHER DEVELOPMENT**

The system can be designed for further enhancement .This could also be developed according to the growing needs of the customer.

**BIBLIOGRAPHY**

* **For Java installation**
  + - https://www.java.com/en/download/
* **For Oracle Database installation**
  + - <http://www.oracle.com/index.html>
* **Reference websites**
  + - [www.javatpoint.com](http://www.javatpoint.com)
    - [www.w3schools.com](http://www.w3schools.com)
    - <http://www.tutorialspoint.com/java/index.htm>
* **Reference Books**
  + - Thinking in java
    - OCJP Certified Programmer for Java
    - Learn Java in Easy Steps
    - Complete reference Java