Mini Project

Banking Management System

Student Name: Kamalesh VB Reg: 12100475
Student Name: Sandhipvigash A T Reg: 12109298

Submitted to
Dr. Om Prakash Yadav (26121)
Asst. Professor,
School of Computer Science and Engineering

LOVELY PROFESSIONAL UNIVERSITY
PHAGWARA, PUNJAB

ABSTRACT:

The banking industry is constantly evolving with technological advancements, and a banking management system is a crucial component for efficient and effective management of banking operations. This system can be used for a wide range of tasks such as managing customer accounts, tracking transactions, generating reports, and providing online banking services. The abstract for a banking management system would outline the system's purpose, the features it offers, and the benefits it provides to both the bank and its customers. The system's architecture, implementation, and maintenance strategies can also be discussed to provide a comprehensive overview. The abstract would highlight the importance of using a banking management system to streamline banking operations, enhance customer satisfaction, and improve overall business efficiency. During the past several decades personnel function has been transformed from a relatively obscure record keeping staff to central and top level management function.A computer based management system is designed to handle all the primary information required to calculate monthly statements of customer account which include monthly statement of any month. Separate database is maintained to handle all the details required for the correct statement calculation and generation.
This project intends to introduce more user friendliness in the various activities such as record updation, maintenance, and searching.

EXISTING SYSTEM:

The existing banking management system is a combination of manual and computerized systems that have evolved over time. In the past, banks relied heavily on manual processes such as paper-based transactions, ledger books, and physical storage of customer data. However, with the advent of computers and internet technology, banks have gradually shifted towards computerized systems for managing their operations. Currently, most banks use a core banking system that is based on centralized servers and databases. These systems are designed to manage customer accounts, process transactions, generate reports, and provide online banking services. The core banking system is usually integrated with various subsystems such as ATM networks, mobile

banking, and internet banking platforms to offer a seamless banking experience to customers. Apart from the core banking system, banks also use various other software applications for specific functions such as loan management, credit card processing, and customer relationship management. These applications are integrated with the core banking system to ensure that all data is centralized and accessible from a single location. Overall, the existing banking management system is a combination of various manual and computerized systems that work together to manage the bank's operations. However, these systems can be complex and expensive to maintain, and banks are constantly exploring new technologies to improve their efficiency and offer better services to customers.

PROPOSED SYSTEM:

A banking management system is a complex software system that is used to manage a bank's various operations, including customer transactions, account management, loans, and other financial services. Here are some key features that could be included in a proposed banking management system:

1. Account Management: The system should allow bank employees to manage customer accounts, including opening and closing accounts, updating customer information, and managing account balances.
2. Transaction Processing: The system should support a wide range of transaction types, including deposits, withdrawals, transfers, and bill payments.
3. Loan Management: The system should be able to manage different types of loans, including personal, business, and mortgage loans. It should allow bank employees to create loan applications, process approvals, manage disbursements, and track repayments.
4. Reporting: The system should provide a range of reports that allow bank managers to monitor key metrics such as deposits, withdrawals, loan balances, and profitability.
5. Security: The system should be designed with a high level of security to protect sensitive customer information and prevent unauthorized access.
6. Integration with other banking systems: The system should be able to integrate with other banking systems, such as ATM networks, online banking systems, and mobile banking apps.
7. Customer Relationship Management: The system should be able to manage customer interactions, including tracking customer inquiries, complaints, and feedback.
8. Compliance: The system should be designed to comply with banking regulations and standards, including anti-money laundering regulations and data privacy laws. Overall, a robust banking management system should be easy to use, reliable, and secure. It should help bank employees to efficiently manage customer accounts, process transactions, and monitor key metrics, while

providing a high level of customer service and support.

ADVANTAGE OF PROJECT:

Increased efficiency: A well-designed banking management system can streamline banking processes, reducing the time and resources required for tasks such as account management, transaction processing, and customer service. Improved customer service: By providing quick and reliable service, a banking management system can enhance the overall customer experience, resulting in increased customer loyalty and retention. Enhanced security: Banking management systems often come equipped with security features such as encryption, access controls, and audit trails, which help to protect sensitive customer information and prevent fraudulent activity. Better decision-making: A banking management system can generate data and reports that provide insight into customer behavior, market trends, and other key performance indicators, allowing banks to make informed decisions about their operations. Increased profitability: By reducing operational costs and improving customer satisfaction, a banking management system can ultimately lead to increased profitability for banks

SYSTEM REQUIREMENT:

Hardware specifications Hardware is a set of physical components, which performs the functions of applying appropriate, predefined instructions. In other words, one can say that electronic and mechanical parts of computer constitute hardware. This package is designed on a powerful programming language Visual Basic. It is a powerful Graphical User Interface. The backend is ACCESS, which is used to maintain database. It can run on almost all the popular microcomputers. The following are the minimum hardware specifications to run this package:

Personal Computer: - It minimum contains P-III Processor with 128 MB RAM Software Requirements: The software is a set of procedures of coded information or a program which when fed into the computer hardware, enables the computer to perform the various tasks. Software is like a current inside the wire, which cannot be seen but its effect can be felt.

1. Operating System:- Windows NT / 2000 / XP
2. Application Software:- Application software uses front end visual basic and database access etc.

Code:

```
import java.io.*;

class BankWork {
  // initialize and declare objects.
  final int max_limit = 20;
  final int min_limit = 1;
  final double min_bal = 500;

  private String name[] = new String[20];
  private int accNo[] = new int[20];
  private String accType[] = new String[20];
  private double balamount[] = new double[20];

  static int totRec = 0;

  // create a constructor here of Bank.
  BankWork() {
    for (int i = 0; i < max_limit; i++) {
      name[i] = "";
      accNo[i] = 0;
      accType[i] = "";
      balamount[i] = 0.0;
    }
  }

  // Create method to create New entry.
  public void newEntry() {
    String str;

    int account;
    double amount;
    boolean permit;
    permit = true;

    if (totRec > max_limit) {
      System.out.println("\n\n\nSorry we cannot admit you in our bank...\n\n\n");
      permit = false;
    }

    // create new entry.
    if (permit = true) {
      totRec++;

      // Incrementing Records
      System.out.println("\n\n\n=====SAVING NEW ENTRY=====");
      try {
        accNo[totRec] = totRec; //Created  AutoNumber  to accNo so no invalid id occurs
        System.out.println("Account Number :  " + accNo[totRec]);

        // create object.
        BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));
```

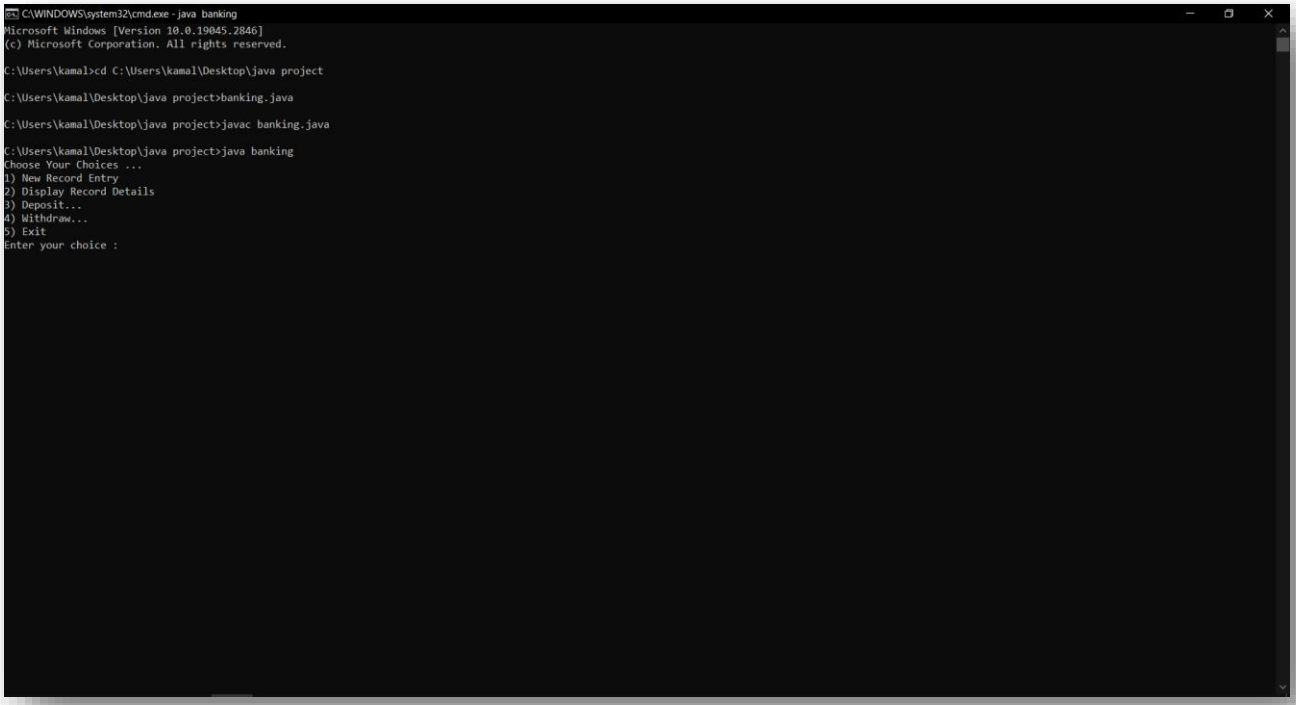
```
      // enter the name of customer here.
      System.out.print("Enter the name of the Customer : ");
      System.out.flush();

      name[totRec] = obj.readLine();

      // enter the type of account.
      System.out.print("Enter Account Type : ");
      System.out.flush();

      accType[totRec] = obj.readLine();
      do {
        // enter the starting amount.
        // minimum amount must be 1000.
        System.out.print("Enter Initial  Amount to be deposited : ");
        System.out.flush();
        str = obj.readLine();

        balamount[totRec] = Double.parseDouble(str);
      }
      while (balamount[totRec] < min_bal);

      System.out.println("\n\n\n");
    } catch (Exception e) {
      System.out.println("Exception in Entering a record.....");
    }
  }
}

// create method to display records.
public void display() {
  String str;
  int account = 0;
  boolean valid = true;

  System.out.println("\n\n=====DISPLAYING THE RECORDS=====\n");
  try {
    // create object.
    BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));

    // enter account number.
    System.out.print("Enter the account number for display record : ");
    System.out.flush();

    str = obj.readLine();
    account = Integer.parseInt(str);

    // check for valid account number
    if (account < min_limit || account > totRec) {
      System.out.println("\n\n\nInvalid Account Number \n\n");
      valid = false;
    }

    if (valid == true) {
      System.out.println("\n\nAccount Number : " + accNo[account]);
```

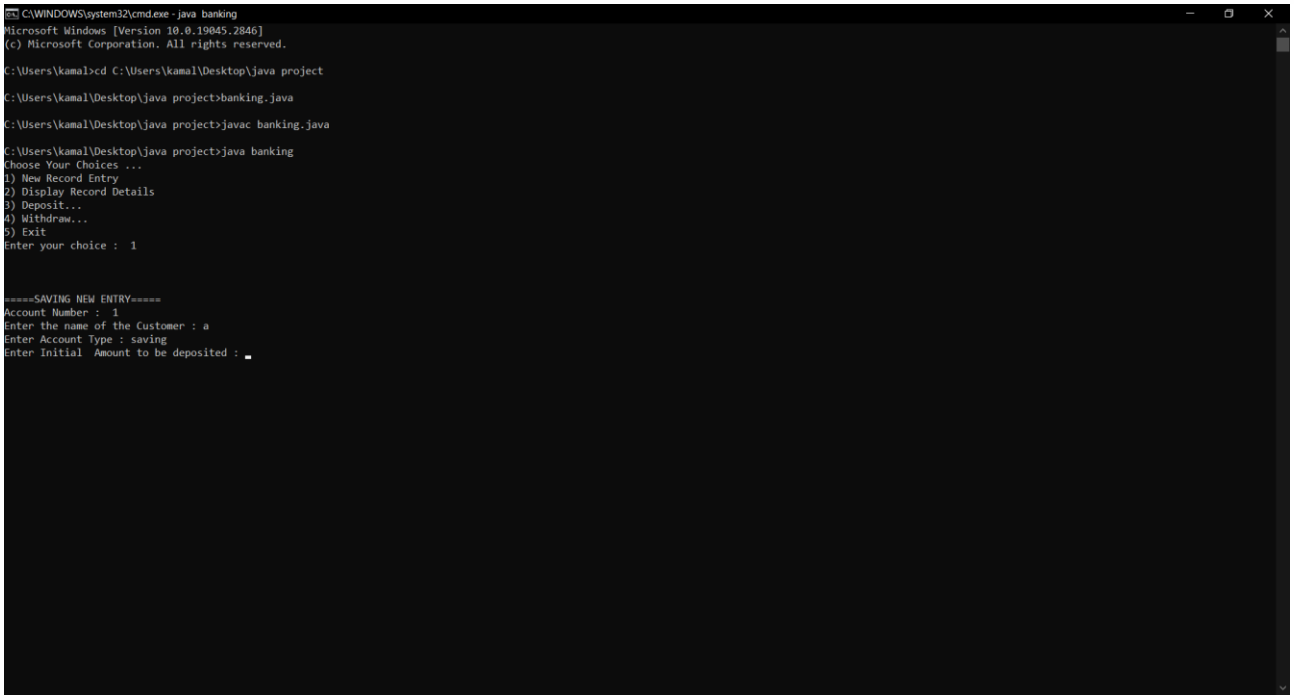
```
      System.out.println("Name : " + name[account]);
      System.out.println("Account Type : " + accType[account]);
      System.out.println("Balance Amount : " + balamount[account] + "\n\n\n");
    }
  } catch (Exception e) {
    System.out.println("Exception in Displaying record.....");
  }
}

// create method to deposit amount.
public void deposit() {
  String str;
  double amount;
  int account;
  boolean valid = true;
  System.out.println("\n\n\n=====DEPOSIT AMOUNT=====");

  try {
    // create object.
    BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));

    System.out.print("Enter Account No : ");
    System.out.flush();

    str = obj.readLine();
    account = Integer.parseInt(str);

    // check valid account number.
    if (account < min_limit || account > totRec) {
      System.out.println("\n\n\nInvalid Account Number \n\n");
      valid = false;
    }

    if (valid == true) {
      System.out.print("Enter Amount you want to Deposit  : ");
      System.out.flush();

      str = obj.readLine();
      amount = Double.parseDouble(str);

      balamount[account] = balamount[account] + amount;

      //Displaying Depsit Details
      System.out.println("\nAfter Updation...");
      System.out.println("Account Number :  " + account);
      System.out.println("Balance Amount :  " + balamount[account] + "\n\n\n");
    }
  } catch (Exception e) {
    System.out.println("Exception in Depositing record.....");
  }
}

// creating method for withdraw money.
public void withdraw() {
  String str;
```

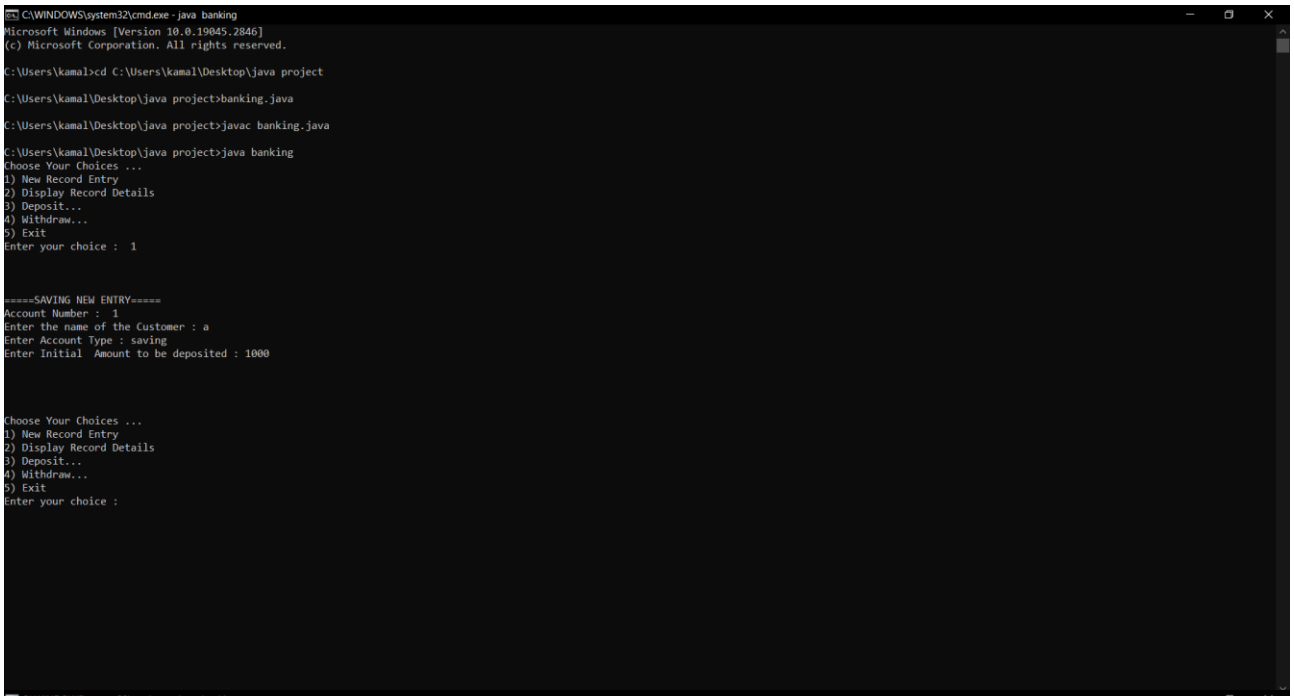
```
    double amount, checkamount;
    int account;
    boolean valid = true;

    System.out.println("\n\n\n=====WITHDRAW MONEY=====");
    try {
      // create object.
      BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));

      // enter account number for entering money
      System.out.print("Enter the account number to deposit money : ");
      System.out.flush();

      str = obj.readLine();
      account = Integer.parseInt(str);

      // check for valid account number.
      if (account < min_limit || account > totRec) {
        System.out.println("\n\n\nInvalid Account Number \n\n");
        valid = false;
      }

      if (valid == true) {
        System.out.println("Balance is : " + balamount[account]);
        System.out.print("Enter Amount you want to withdraw  : ");
        System.out.flush();

        str = obj.readLine();
        amount = Double.parseDouble(str);
        checkamount = balamount[account] - amount;

        if (checkamount >= min_bal) {
          balamount[account] = checkamount;

          // Updating the amount after withdraw.
          System.out.println("\nAfter Updation...");
          System.out.println("Account Number :  " + account);
          System.out.println("Balance Amount :  " + balamount[account] + "\n\n\n");
        } else {
          System.out.println("\n\nAs per Bank Rule you should maintain minimum balance of Rs
500\n\n\n");
        }
      }
    } catch (Exception e) {
      System.out.println("Exception in Withdrawing record.....");
    }
  }
};

public class banking {
  public static void main(String args[]) {
    String str;
    int choice;
    choice = 0;
```

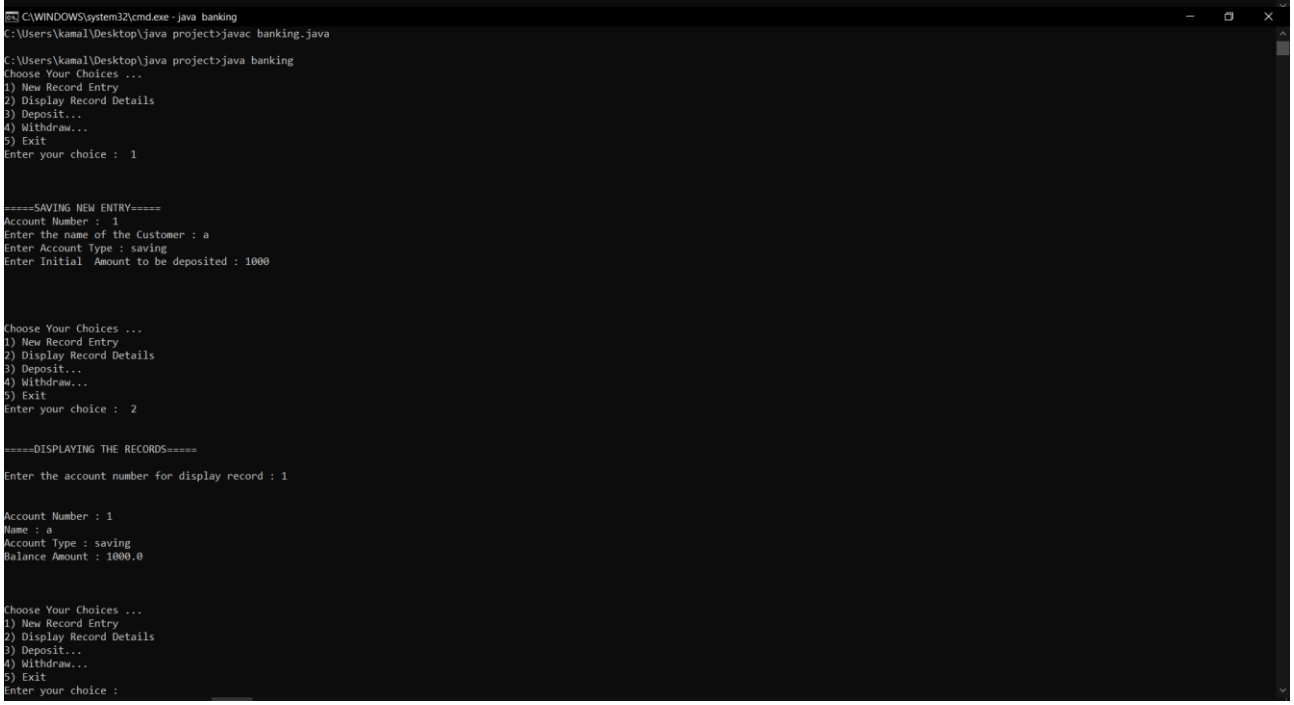
```
    BankWork BW_obj = new BankWork();

    do {
      // creating Menu.
      System.out.println("Choose Your Choices ...");
      System.out.println("1) New Record Entry ");
      System.out.println("2) Display Record Details ");
      System.out.println("3) Deposit...");
      System.out.println("4) Withdraw...");
      System.out.println("5) Exit");
      System.out.print("Enter your choice :  ");
      System.out.flush();
      try {
        // creating objects.
        BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));
        str = obj.readLine();
        choice = Integer.parseInt(str);

        switch (choice) {
        case 1:
          // for new entry.
          BW_obj.newEntry();
          break;

        case 2:
          // for display.
          BW_obj.display();
          break;

        case 3:
          // for deposit.
          BW_obj.deposit();
          break;

        case 4:
          // for display.
          BW_obj.withdraw();
          break;

        case 5:
          System.out.println("\n\n.....THANKS FOR VISITING.....");
          break;

        default:
          System.out.println("\nInvalid Choice \n\n");
        }
      } catch (Exception e) {
        System.out.println("Exception in Main....");
      }
    }
    while (choice != 5);
  }
}
```

Snap of project:






C:\WINDOWS\system32\cmd.exe - java banking
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.
C:\Users\kamal>cd C:\Users\kamal\Desktop\java project
C:\Users\kamal\Desktop\java project>banking.java
C:\Users\kamal\Desktop\java project>javac banking.java
C:\Users\kamal\Desktop\java project>java banking
Choose Your Choices ...
1) New Record Entry
2) Display Record Details
3) Deposit...
4) Withdraw...
5) Exit
Enter your choice : 1
=====SAVING NEW ENTRY=====
Account Number : 1
Enter the name of the Customer : a
Enter Account Type : saving
Enter Initial Amount to be deposited : 1000
Choose Your Choices ...
1) New Record Entry
2) Display Record Details
3) Deposit...
4) Withdraw...
5) Exit
Enter your choice :


C:\WINDOWS\system32\cmd.exe - java banking
C:\Users\kamal\Desktop\java project>javac banking.java
C:\Users\kamal\Desktop\java project>java banking
Choose Your Choices ...
1) New Record Entry
2) Display Record Details
3) Deposit...
4) Withdraw...
5) Exit
Enter your choice : 1
=====SAVING NEW ENTRY=====
Account Number : 1
Enter the name of the Customer : a
Enter Account Type : saving
Enter Initial Amount to be deposited : 1000
Choose Your Choices ...
1) New Record Entry
2) Display Record Details
3) Deposit...
4) Withdraw...
5) Exit
Enter your choice : 2
=====DISPLAYING THE RECORDS=====
Enter the account number for display record : 1
Account Number : 1
Name : a
Account Type : saving
Balance Amount : 1000.0
Choose Your Choices ...
1) New Record Entry
2) Display Record Details
3) Deposit...
4) Withdraw...
5) Exit
Enter your choice :

Conclusion:

The main objectives of a banking system are to provide financial services to customers, such as deposit accounts, loans, and payment services. These services are designed to help individuals and businesses manage their finances effectively and efficiently. The specific objectives of a banking system can vary depending on the organization, but typically include:

1. Safeguarding customer deposits and financial information through secure systems and protocols.
2. Providing a range of financial products and services to meet the needs of customers.
3. Offering competitive interest rates and fees to attract and retain customers.
4. Efficiently processing transactions and managing customer accounts .
5. Developing and implementing effective risk management strategies to protect the bank and its customers from financial losses.
6. Adhering to regulatory requirements and maintaining compliance with banking laws and regulations.
7. Delivering excellent customer service to enhance the customer experience and build customer loyalty.

Overall, the main objective of a banking system is to provide financial stability and support to individuals and businesses while ensuring the safety and security of their assets.

Future Enhancement:

There are several potential future enhancements that could be made to a banking system to improve its functionality and provide a better customer experience. Some possible enhancements include:

Mobile Banking: Develop a mobile application that allows customers to access their accounts, make transactions, and perform other banking activities from their smartphones or tablets.

Personalized Services: Implement personalized services that use customer data to offer customized financial products and services that meet individual needs and preferences.

Artificial Intelligence: Utilize artificial intelligence and machine learning to enhance fraud detection, improve risk management, and provide personalized financial advice to customers.

Blockchain Technology: Explore the potential benefits of blockchain technology for secure, transparent, and efficient transactions and payment processing.

Biometric Authentication: Implement biometric authentication methods such as facial recognition or fingerprint scanning to enhance security and streamline the authentication process for customers. Chatbots and Virtual Assistants: Use chatbots and virtual assistants to provide 24/7 customer support, answer frequently asked questions, and assist with routine banking tasks.

Open Banking: Explore the potential benefits of open banking, which allows customers to share their banking data with third-party service providers to access innovative financial products and services. Overall, these enhancements could help banks to remain competitive in the rapidly evolving financial industry while providing a better customer experience and enhancing security and efficiency.

References:

1.Java Programming for Banking by Kartik Bhatnagar and Amit. Bhatnagar
2.Banking and Finance on the Internet by Mary J. Cronin.
3.Java in Finance: Develop and Implement Financial Applications by Kumar and Sandeep.

4.Security in Online Banking by Jeffrey Macklin and Jeffrey Lee
5.Java-Based Internet Banking System by Li Dan
6.A Comparative Study of Banking Systems in Java by Ghodrati.