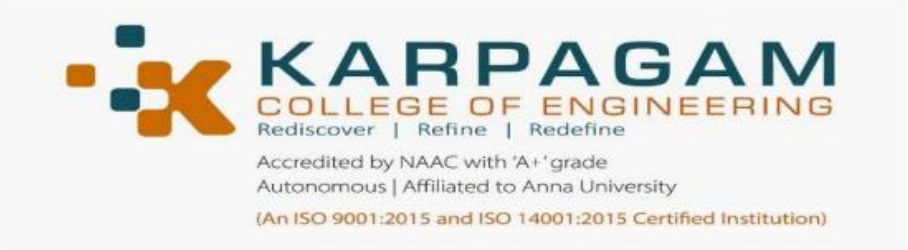


# ELECTRICAL AND ELECTRONICS ENGINEERING

## MINI BANKING APPLICATION

### 21PE04 & ADVANCED JAVA PROGRAMMING

**MINI PROJECT REPORT**

**KARTHICK S K**

**(717821E131)**

**VISVANTH V**

**(717821E162)**

# TABLE OF CONTENTS

## ABSTRACT:

In any Bank Transaction, there are several parties involved to process transaction like a merchant, bank, receiver, etc. so there are several numbers reasons that transaction may get failed, declined, so to handle a transaction in Java, there is a JDBC
(Java Database Connectivity) which provides us an API to connect, execute, fetch

data from any databases. It provides the language Java database connectivity standards. It is used to write programs required to access databases.

Transactions in JDBC provide us a feature that considers a complete SQL statement As one unit, then executes once, and if any statement fails, the entire transaction fails.
To use transaction, we have to set **setAutoCommit(false);** manually, and once all the statements are executed successfully, making changes in the database's **commit()** methodwill be required.

In this Mini Banking Application, to handle a transaction, we are using JDBC Transaction to make transactions consistent. This Application Provides Menu-Driven Console Interface to a User Using that User can perform functions like create Account, Login, View Balance And Transfer Money To The Other Customer.

## INTRODUCTION:

The purpose of this report is to provide an overview of a mini banking application that has been developed for managing financial transactions. The mini banking application is a software system designed to simplify the process of managing financial transactions for individuals and small businesses. This report will outline the key features of the application, including user interface, functionality, and security features. Additionally, this report will discuss the benefits of using the mini banking application, including increased efficiency, improved accuracy, and enhanced security. Finally, this report will highlight some potential areas for Future development and improvement.

## PROJECT DESCRIPTION:

The mini banking application is a software system designed to simplify the process of managing financial transactions for individuals and small businesses. The application allows users to create and manage their accounts, deposit and withdraw money, transfer funds between accounts, and view their transaction history.The user interface of the mini banking application is intuitive and easy to navigate. Users can access their accounts through a login screen, which requires a username and password for authentication. Once logged in, users can view their account balance and make transactions using a simple and user-friendly interface.The functionality of the mini banking application includes the ability to deposit and withdraw money, transfer funds between accounts, and view transaction history. The application also includes features such as account statements, which allow users to view their transaction history in a detailed and organized manner.The mini banking application also includes several security features to protect users' financial information. These security features include encryption of sensitive data, such as passwords and account information, as well as multi-factor authentication to prevent unauthorized access to user accounts.The benefits of using the mini banking application include increased efficiency, improved accuracy, and enhanced security. By automating many of the financial transactions that were previously done manually, users can save time and reduce errors. Additionally, the application's security features ensure that users' financial information is protected from unauthorized access and theft.Some potential areas for future development and improvement of the mini banking application include the addition of new features, such as bill payment and investment management, as well as further enhancements to the application's security features to keep pace with evolving threats to cybersecurity.

## WORKING:

The mini banking application works by providing users with a secure and convenient platform for managing their financial transactions. The application is designed to automate many of the processes involved in managing financial accounts, thereby increasing efficiency and reducing the potential for errors.When a user logs into the mini banking application, they are presented with a dashboard that displays their account information, including their account balance, transaction history, and other relevant information. From this dashboard, users can initiate a variety of financial transactions, such as depositing or withdrawing money from their account, transferring funds between accounts, or making payments to other accounts.To initiate a transaction, the user simply selects the appropriate option from the dashboard and enters the necessary details, such as the amount of money transferred, the recipient's account information, or the purpose of the transaction. The application then processes the transaction and updates the user's account information accordingly.The mini banking application also includes a variety of security features to protect users' financial information. These features include encryption of sensitive data, such as passwords and account information, as well as multi-factor authentication to prevent unauthorized access to user accounts. Additionally, the application employs a variety of monitoring and detection tools to identify and prevent potential security breaches.Overall, the mini banking application streamlines the process of managing financial transactions for individuals and small businesses, while also providing a secure and convenient platform for accessing and managing financial accounts.

## 2. REQUIREMENTS SPECIFICATION:

### 2.1 EXTERNAL INTERFACE REQUIREMENTS:

#### Hardware Requirements:

- **Minimum Ram:** 1GB
- **Hard disk:** 128GB
- **Processor:** Intel core i5

#### Software Requirements:

- **Operating system:**Windows 11
- **Front design:**eclipse
- **Front-End Language:**JAVA Programming
- **Back-End:** MY SQL
- **Back-End Connectivity:**JDBC

## 2.2. FUNCTIONAL REQUIREMENTS SPECIFICATION:

JDBC involves following functions are:

- Provide a consistent and standard way to access and interact with relational databases.
- Support connection pooling to improve performance reduce the overhead of creating andclosing connections
- Support for prepared statements to help prevent SQL injection attacks and improve performance.
- Provide support for both simple and complex SQL queries, including SELECT, INSERT,UPDATE, and DELETE statements.
- Allow for the retrieval of result sets with support for data types including strings, integers, and dates.
- Provide transaction support, including commit and rollback operations, to ensure data integrity
- Handle errors and exceptions that may occur during database operations.

## 3.HIGH LEVEL DESIGN:

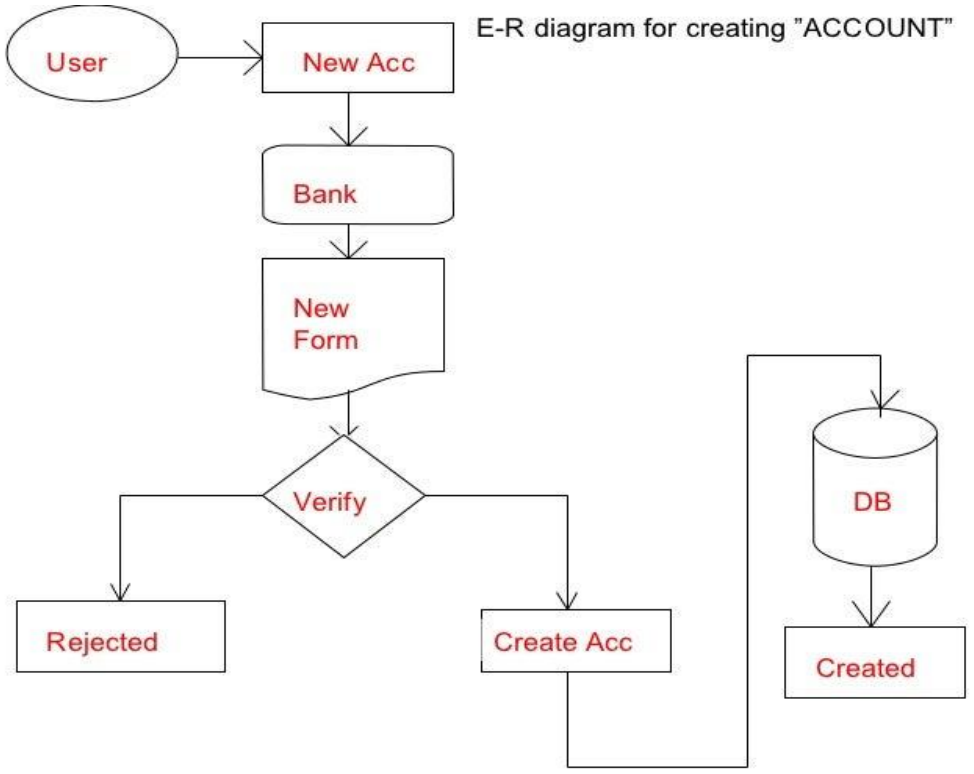


**Fig 3.1. Dataflow Diagram for mini banking application**

This data flow diagram shows the connectivity between the user and the report produced over here. When th data is entered in the system it will get stored in database which is easy to retrieve the data for the future purpose. This system is very useful in HR. This flow diagram is easy to understand the relation between the user and Data base.

## PROJECT IMPLEMENTATION:

**Requirements Gathering:** This involves gathering requirements from stakeholders, such as end- users, management, and IT staff. This process involves identifying the features and functionalities required by the application.

**Design:** Based on the requirements gathered, a design document is created that outlines the architecture, user interface, and database schema. This document will serve as the blueprint for the development of the application.

**Development**: The actual development of the application begins at this stage. The development process involves coding the application, integrating it with third-party services, and testing the application to ensure it meets the requirements.

**Testing:** The application is tested to ensure that it meets the functional and non-functional requirements, such as performance, security, and reliability.

**Deployment:** Once the application is tested and deemed ready for production use, deploy it to a production environment, such as a web server or cloud hosting service.

**User Acceptance Testing (UAT):** Conduct UAT with real users to validate the application's usability, functionality, and user experience. Gather feedback from users and make any necessary adjustments to address their concerns or suggestions.

**Documentation:** Create comprehensive documentation for the application, including user manuals, installation guides, and technical documentation. Document the application's functionalities, configurations, and maintenance procedures to assist with future updates or maintenance.

**Deployment to Production:** Once the application has passed UAT and all necessary adjustments have been made, deploy the final version to the production environment and make it available to users for regular use.

## SOURCE CODE:

Step 1: Create Database name bank

Step 2: Create Table name customer

```
// Create a database
CREATE DATABASE BANK;
// Create table
CREATE TABLE customer (
ac_no int NOT NULL AUTO_INCREMENT,
cname varchar(45) DEFAULT NULL,
balance varchar(45) DEFAULT NULL,
pass_code int DEFAULT NULL,
PRIMARY KEY (ac_no),
UNIQUE KEY cname_UNIQUE (cname)
) ;
package banking;
import java.sql.Connection;
import java.sql.DriverManager;
// Global connection Class
public class connection {
static Connection con; // Global Connection Object
public static Connection getConnection()
{
try {
String mysqlJDBCDriver = "com.mysql.cj.jdbc.Driver"; //jdbc driver
String url= "jdbc:mysql://localhost:3306/mydata"; //mysql url
String user = "root"; //mysql username
String pass = "Pritesh4@"; //mysql passcode
Class.forName(mysqlJDBCDriver);
con = DriverManager.getConnection(url, user,pass);
}
catch (Exception e) {
System.out.println("Connection Failed!");}
17
}
```

```
package banking;
return con;
}
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.SQLIntegrityConstraintViolationException;
import java.sql.Statement;
public class bankManagement {
private static final int NULL = 0;
static Connection con = connection.getConnection();
static String sql = "";
public static boolean
createAccount(String name,int passCode) // create account function
{
try {
if (name == ""  passCode == NULL) {
System.out.println("All Field Required!");
return false;
}
Statement st = con.createStatement();
sql = "INSERT INTO customer(cname,balance,pass_code) values('"
+ name + "',1000," + passCode + ")";
if (st.executeUpdate(sql) == 1) {
System.out.println(name+ ", Now You Login!");
return true;
}
}
catch (SQLIntegrityConstraintViolationException e) {
System.out.println("Username Not Available!");
18
}
```

```
catch (Exception e) {
e.printStackTrace();
}
return false;
}
public static boolean
loginAccount(String name, int passCode) // login method
{
try {
if (name == ""  passCode == NULL) {
System.out.println("All Field Required!");
return false;
}
sql = "select * from customer where cname='"
+ name + "' and pass_code=" + passCode;
PreparedStatement st= con.prepareStatement(sql);
ResultSet rs = st.executeQuery();
BufferedReader sc = new BufferedReader(new InputStreamReader(System.in));
if (rs.next()) {
int ch = 5;
int amt = 0;
int senderAc = rs.getInt("ac_no");
int receiveAc;
while (true) {
try {
System.out.println("Hallo, "+ rs.getString("cname"));
System.out.println("1)Transfer Money");
System.out.println("2)View Balance");
System.out.println("5)LogOut");
System.out.print("Enter Choice:");
ch = Integer.parseInt(sc.readLine());
if (ch == 1) {
System.out.print("Enter Receiver A/c No:");
receiveAc = Integer.parseInt(sc.readLine());
19
System.out.print("Enter Amount:");
```

```
amt = Integer.parseInt(sc.readLine());
if (bankManagement.transferMoney(senderAc, receiveAc,amt)) {
System.out.println("MSG : Money Sent Successfully!\n");
}
else {
System.out.println("ERR : Failed!\n");
}
}
else if (ch == 2) {
bankManagement.getBalance(senderAc);
}
else if (ch == 5) {
break;
}
else {
System.out.println("Err : Enter Valid input!\n");
}
}
catch (Exception e) {
e.printStackTrace();
}
}
}
else {
return false;
}
return true;
}
catch (SQLIntegrityConstraintViolationException e) {
System.out.println("Username Not Available!");
}
catch (Exception e) {
e.printStackTrace();
}
20
return false;
```

```
}
public static void getBalance(int acNo)
{
try {
sql = "select * from customer where ac_no="+ acNo;
PreparedStatement st = con.prepareStatement(sql);
ResultSet rs = st.executeQuery(sql);
System.out.println(" ");
System.out.printf("%12s %10s %10s\n","Account No", "Name","Balance");
while (rs.next()) {
System.out.printf("%12d %10s %10d.00\n",rs.getInt("ac_no"),rs.getString("cname"),
rs.getInt("balance"));
}
System.out.println(" \n");
}
catch (Exception e) {
e.printStackTrace();
}
}
public static boolean transferMoney(int sender_ac,int reveiver_ac,int amount) throws SQLException
{
System.out.println("1)Create Account");
System.out.println("2)Login Account");
try {
System.out.print("\n Enter Input:");
ch = Integer.parseInt(sc.readLine());
switch (ch) {
case 1:
try {
System.out.print("Enter Unique UserName:");
name = sc.readLine();
System.out.print("Enter New Password:");
pass_code = Integer.parseInt(sc.readLine());
if (bankManagement.createAccount(name, pass_code)) {
System.out.println("MSG : Account Created Successfully!\n");
}
```

```
else {
System.out.println("ERR : Account Creation Failed!\n");
}
}
catch (Exception e) {
System.out.println(" ERR : Enter Valid Data::Insertion Failed!\n");
}
Break;
case 2:
try {
System.out.print("Enter UserName:");
name = sc.readLine();
System.out.print("Enter Password:");
pass_code = Integer.parseInt(
sc.readLine());
if (bankManagement.loginAccount(name, pass_code)) {
System.out.println("MSG : Logout Successfully!\n");
}
else {
System.out.println("ERR : login Failed!\n");
23
}
}
catch (Exception e) {
System.out.println(" ERR : Enter Valid Data::Login Failed!\n");
}
break;
default:
}
System.out.println("Invalid Entry!\n");
if (ch == 5) {
System.out.println("Exited Successfully!\n\n Thank You :)");
break;
}
}
catch (Exception e) {System.out.println("Enter Valid Entry!") }}
```

**OUTPUT:**

**Login:**

```
   ->||    Welcome to InBank    ||<-

     1)Create Account
     2)Login Account

     Enter Input:2
 Enter  UserName:pritesh
 Enter  Password:123
```

**View Balance:**

```
Enter  UserName:pritesh
Enter  Password:123
Hallo, pritesh
1)Transfer Money
2)View Balance
5)LogOut
Enter Choice:2
------------------------------------------------------------
  Account No       Name    Balance
         112    pritesh          0.00
------------------------------------------------------------
```

**Transfer Money**:

```
Hallo, pritesh
1)Transfer Money
2)View Balance
5)LogOut
Enter Choice:1
Enter Receiver  A/c No:110
Enter Amount:5000
Insufficient Balance!
ERR :  Failed!
```

**Logout:**

```
Hallo, pritesh
1)Transfer Money
2)View Balance
5)LogOut
Enter Choice:5
MSG : Logout Successfully!
```

## CONCLUSION:

The project is aimed at exposing the relevance Of The Mini banking application. This Application Provides Menu-Driven Console Interface to a User Using that User can perform functions like create Account, Login, View Balance And Transfer Money To The Other Customer.