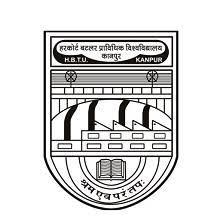
***Bank Management System***



***HARCOURT BUTLER TECHNICAL UNIVERSITY SESSION 2021-22***

Submitted to: Submitted by:

Prof. Raghuraj Singh Sir , Harsh Prajapati (31)

Mr. Amlendra Kumar Goldee Arora (27)

Ujjwal Singh (62)

Utkarsh Dixit (64)

Nitin Kumar (42)

Certificate

***This is to certify that the project on Bank Management System is***

**Submitted By**

**Goldee Arora- 200104027**

**Harsh Prajapati – 200104031**

**Nitin Kumar – 200104042**

**Ujjwal Singh – 200104062**

**Utkarsh Dixit – 200104064**

***Of IV semester Computer Science & Engineering in the year 2022 for the subject Software Engineering***

**ACKNOWLEDGEMENT**

It is my privilege to offer special veneration and heartiest gratitude to our esteemed guides***, Prof. Amlendra Kumar Sir*** & ***Prof. Priyanka Pandey*** **Ma’am**, without whose encourage attitude, sagacious, guidance and constant relevant help this project would never have come out with flying colours. We extend our sincere thanks to our respected Head of the department, ***Prof. Raghuraj Singh*** ***Sir***, for allowing us to use the facilities available. We would like to thank the other faculty members also, for the support and encouragement they have given us during the course of the work.

**Abstract**

The Bank Account Management System is an application for maintaining a person's account in a bank. In this project I tried to show the working of a banking account system and cover the basic functionality of a Bank Account Management System. To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks.

The main aim of this project is to develop software for Bank Account Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software. This project is developed using Java language. Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer, while proving compliance and staying on the schedule and within budget.

Aim of the Project

The main aim of designing and developing this Internet banking System Java primarily based on

Engineering project is to provide secure and efficient net banking facilities to the banking customers over the internet. Apache Server Pages, SQLite database used to develop this bank application where all banking customers can login by their account login id and password. Users will have all options and features like deposit, withdrawal and others.

***Project Profile***

***Project Title***

* **Bank Management System**

HARDWARE AND SOFTWARE USED:-

SOFTWARE REQUIREMENT:-

Operating System: -

* Windows Xp/7/8/10/, Mac-os, Linux-8

Front end: - Microsoft Visual Studio

Back end: - SQLite

HARDWARE REQUIREMENTS:-

RAM: -

* 512 MB or more

Hard disk: -

* 10 GB or more

DOCUMENTATION TOOLS :-

* Microsoft word 2019

***Software Development Model***

A software-process-model is a brief representation of how a software process works. It helps to develop a software as per under the functionality of the user keeps a sequence of steps followed for a good software. A process model is chosen based on the nature of the project and application, the methods and tools to be used for functionalities that are required as per user. For the Bank Management system we choose spiral model

***Spiral Modal***

The spiral model is a very beneficial and efficient model. It is the combination of the prototyping model and the waterfall model. The spiral model is mostly used for large, Risky, expensive, and complicated projects as best suits for Bank Management system. A big advantage which is risk management is the highlight of this model, it is one of the best SDLC model available for the risk analysis and risk management at each part. User requirements can be changed at later phases so it is flexible as well. The steps in the spiral lifecycle model are follows:-

1. The system requirements are gathered in detailed from users. It can be done with gathering information from a number of users.

2. An elementary design is created for the system. It is the most important phase of ‘Spiral Model’. To create a cost effective project, strategies are applied during creation and alternatives are found.

This stage assists with recognizing and resolve all the potential risks in the undertaking development. If any uncertain functionality meets in requirements, with the sufficient data software proceeds, changes are made to meet and remove the uncertain data. With the help of the preliminary design first prototype of new system is developed. A second prototype is evolved as follows:

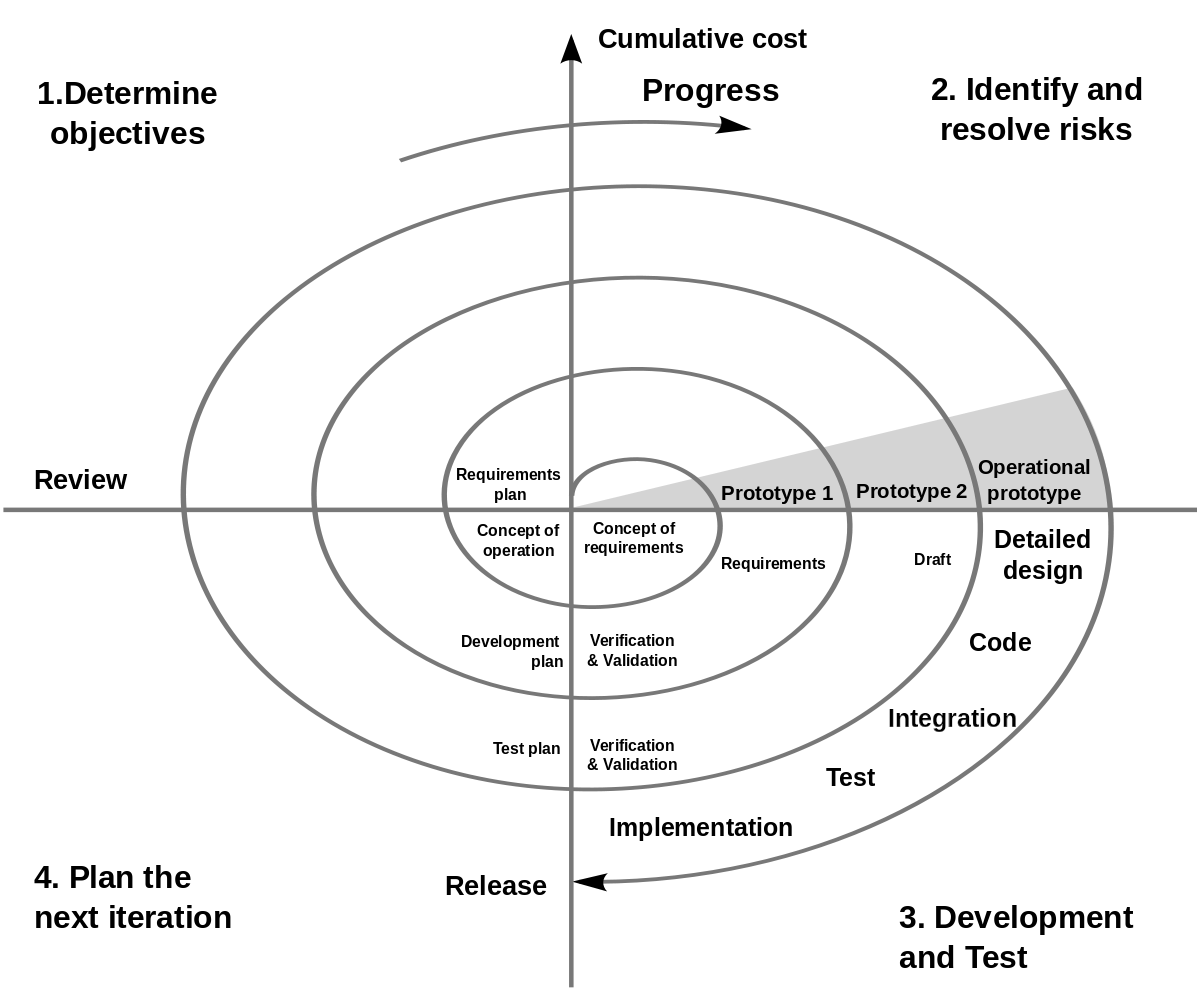
1. Evaluate the strength weakness and risks of first prototype.

2. Defining requirements for the second prototype

3. Plan and design for the second prototype.

4. Develop and test the second prototype.

Spiral Model



**SRS FOR BANK MANAGEMENT SYSTEM**

Table of Contents -:

1.Introduction

1.1 Purpose

1.2 Scope

1.3 Overview

1.4 Additional Information

2. General Description

3. Functional specifications

3.1Login

3.2Validation

3.3Payment of money

3.4Transfer of money

3.5Transaction report

4. Interface Requirements

4.1 CLI

4.2 Hardware Interface

4.3 Software Interface

5. Performance Requirements

6. Constraints

7 Performance

7.1Security

7.2 Reliability

7.3 Availability

7.4 Maintainability

7.5 Reusability

1. Introduction

This document gives detailed functional and nonfunctional requirements for the

bank management system. This product will support online banking transaction.

The purpose of this document is that the requirements mentioned in it should be

utilized by software developer to implement the system.

* 1. Purpose

Online banking system provides is specifically developed for internet banking for

Balance Enquiry, Funds Transfer to another account in the same bank, Request for

cheque book/change of address/stop payment of cheques, Mini statements (Viewing

Monthly and annual statements).

The Traditional way of maintaining details of a user in a bank was to enter the

details and record them. Every time the user need to perform some transactions he

has to go to bank and perform the necessary actions, which may not be so feasible all the time. It may be a hard-hitting task for the users and the bankers too. The

project gives real life understanding of Internet banking and activities performed by

various roles in the supply chain. Here, we provide an automation for banking

system through Internet. Internet banking system project captures

activities performed by different roles in real life banking which provides enhanced

techniques for maintaining the required in- formation up-to-date, which results in

efficiency. The project gives real life understanding of Internet banking and activities

performed by various roles in the supply chain

* 1. Scope

This Product will automate of banking transaction process.

This Project investigatesthe entry threshold for providing a new transaction service channel via thereal options approach, where the entry threshold is established by using an Internet banking system designed for

the use of normal users(individuals), Industrialists, Entrepreneurs, Educational Institutions(Financial sections),

Organizations and Academicians under transaction rate uncertainty.

1.3 Overview

The system provides easy solution to banks.Overview: The SRS will include two sections, namely:Overall Description:

This section will describe major components of the system,

interconnections, and external interfaces.Specific Requirements: This section will describe the functions of actors, their roles in the system and the constraints faced by sys- tem.

1. General description
   1. Product Perspective:

The client will have client interface in which he can interact with the banking sys- tem. It is a web based interface which will be the web page of the banking application. Starting a page is displayed asking the type of customer he is whether ordinary or a corporate customer. Then the page is redirected to login page where the user can enter the login details. If the login particulars are valid then the user is taken to a home page where he has the entire transaction list that he can perform with the bank. All the above activities come under the client interface.The administrator will have an administrative in- terface which is a GUI so thathe can view the entire system. He will also have a login page where he can enterthe login particulars so that he can perform all his actions. This administrative interface provides different environment such that he can maintain data- base & provide backups for the information in the database. He can register the users by providing them with username, password & by creating account in the database. He can view the cheque book request & perform action to issue the cheque books to the clients.2.2

1. Functional Specifications

This section provides the functional overview of the product. The project will require the PHP as a front end and at the back end the database MYSQL will be running. Various functional modules that can be implemented by the product will be

1. Login

2. Validation

3. Get balance information

4. Withdrawal of money

5. Transfer Money

6. Customer info.

3.1 Login:

Customer logins by entering customer name & a login pin.

3.2 Validation:

When a customer enters the ATM card, its validity must be ensured. Then customer is allowed to enter the valid PIN. The validation can be for following conditionsValidation for lost or stolen cardWhen card is already reported as lost or stolenthen the message “Lost/Stolen card!!!”.Validation for card’s expiry dateIf the card inserted by the customer has crossed the expiry date then the system will prompt

“Expired Card”.Validation for PINAfter validating the card, the validity of PIN must be ensured. If he/she fails to enter valid code for three times then the card will not be returned to him. That means the account can be locked. The counter for number of logins must be maintainedGet balance information:This system must be networked to the bank’s computer. The updateddatabase of every customer is maintained with bank. Hence the balance informationof every account is available in the database and can be displayed to the customer.

3.3 Payment of Money:

A customer is allowed to enter the amount which he/she wishes to withdraw. If the entered amount is less than the available balance and if after withdraw if the minimum required balance is maintained then allow the transaction.

3.4 Transfer of Money:

The customer can deposit or transfer the desired amount of money.

3.5 Transaction Report:

The bank statement showing credit and debit information ofcorresponding account must be printed by the machine.

3.6 Technical Issues

This product will work on client-server architecture. It will require an internet server and which will be able to run PHP applications. The product should support some commonly used browsers such as Internet Explorer, Mozilla Firefox.

Interface Requirements

* 1. CLI

This interface may be intuitive or interactive because in this case there will be an assistance for the user who is operating the System. At most of the places help deskshould be provided for users convenience. The screens appearing should be designed in such a manner that it can draw User attaraction towards the new plans for the customers.Also the pin and password confidentiality should be maintained,This can be done by using asterisks at the password panel.Proper security messages should be displayed at most of the places.

* 1. Hardware Interface

Various interfaces for the product could be

1. Touch screen/Monitor

2. Keypad

3. Continuous battery backup4. Printer which can produce the hard copy.

4. Interface that connects the device to bank’s computer.

5. An interface that can count currency notes.

4.3 Software Interface

1. Any windows operating system.
2. The PHP must be installed. For the database handling MYSQL must be installed. These products are open source products.
3. The final application must be packaged in a set up program, so that the products can be easily installed on machines. This application must be networked to corresponding banks.

4. Performance Requirements

The system should be compatible enough to hold the general traffic .It should not get hang or show some other problems arising out due to large no of concurrent users .

1. The system should be fast enough to meet the customer The high and low temperature should not affect the performance ofthe device. An uninterrupted transaction must be performed.

6.Constraints\*

The information of all the users must be stored in a database that is accessible by the On- lineBanking System.\* The Online Banking System is connected to the computer and is running all 24hours a day.\* The users access the Online Banking System from any computer that has Internet browsing capabilities and an Internet connection.\*The users must have their correct usernames and passwords to enter into the Online Banking System. Design Constraints:

\* Software Language Used :

The languages that shall be used for coding Online Banking System are c , c++ , java , and HTML. For working on the coding phase of the Online job portal. System Web Sphere Application Server/WebSphere Application Server CEServer needs to be installed.

\*Database design :

In our database design, we give names to data flows, processes and data stores. Although the names are descriptive of data, they do not give details .So following DFD, our interest is to build some details of the contents of data flows, processes and data store. A data dictionary is a structured repository of data about data .It is aset of rigorous definitions of all DFD data elements and data structures .

7. Performance

7.1 Security

The banking system must be fully accessible to only authentic user.It should require pin for entry to a new environment.

7.2 Reliability

The application should be highly reliable and it should generate all the updated information in correct order.

7.3 Availability

Any information about the account should be quickly available from any computer

to the authorized user. The previously visited customer’s data must not be cleared.

7.4 Maintainability

The application should be maintainable in such a manner that if any new

requirement occurs then it should be easily incorporated in an individual module.

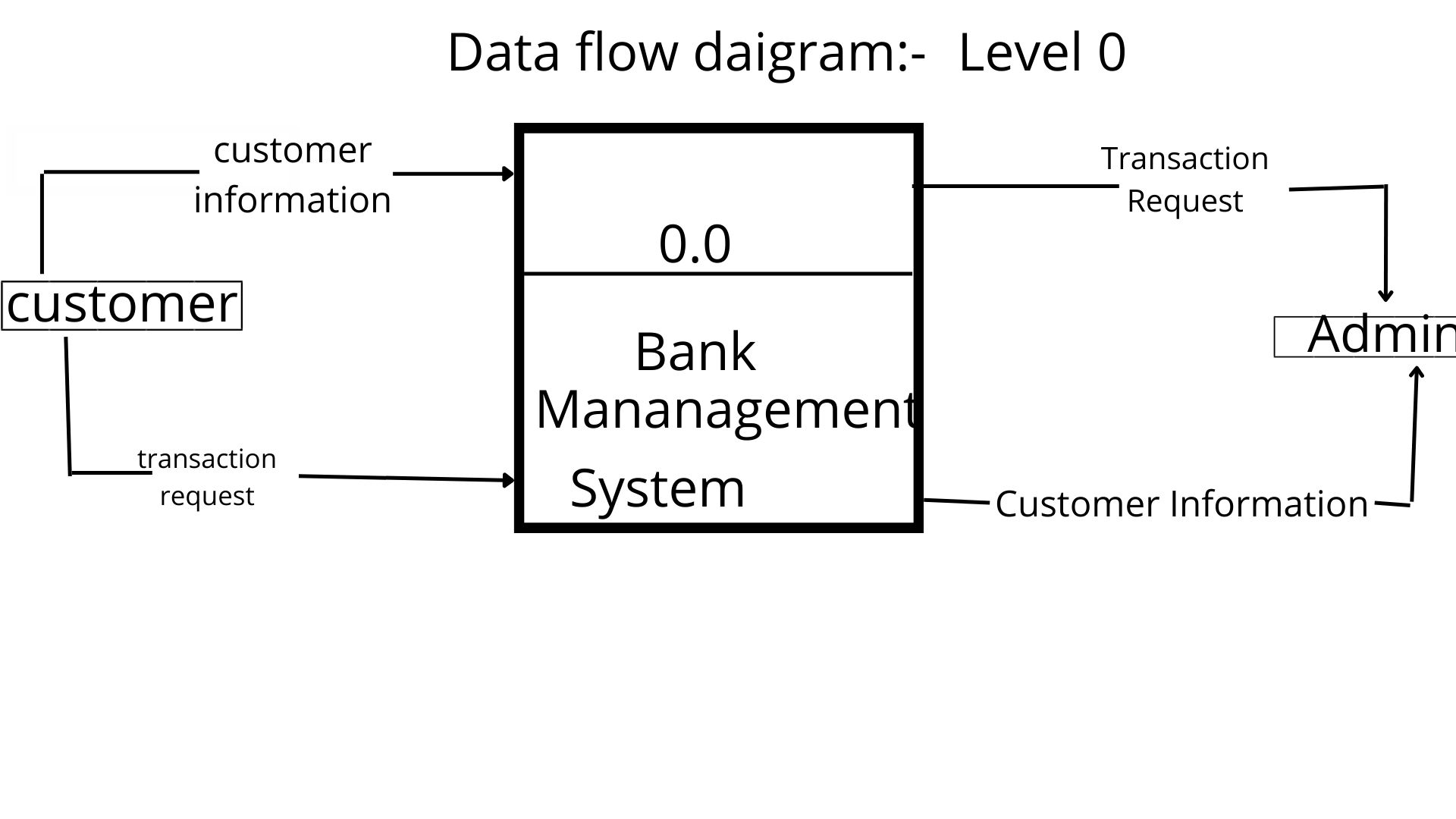
7.5 Portability

The application should be portable on any windows based system. It should not be

machine specific.

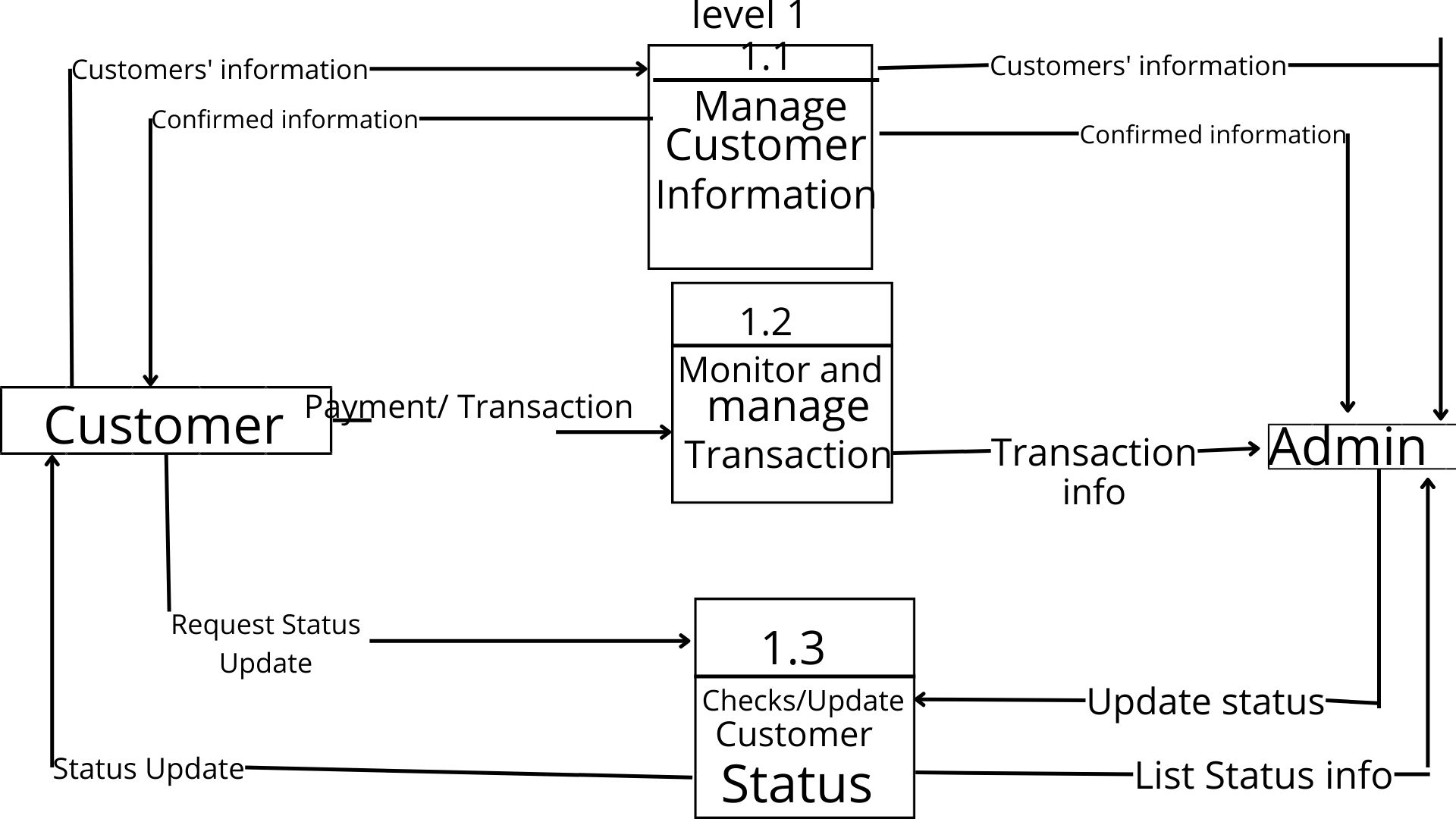
**Data Flow Diagram**

*Level 0 -*

****

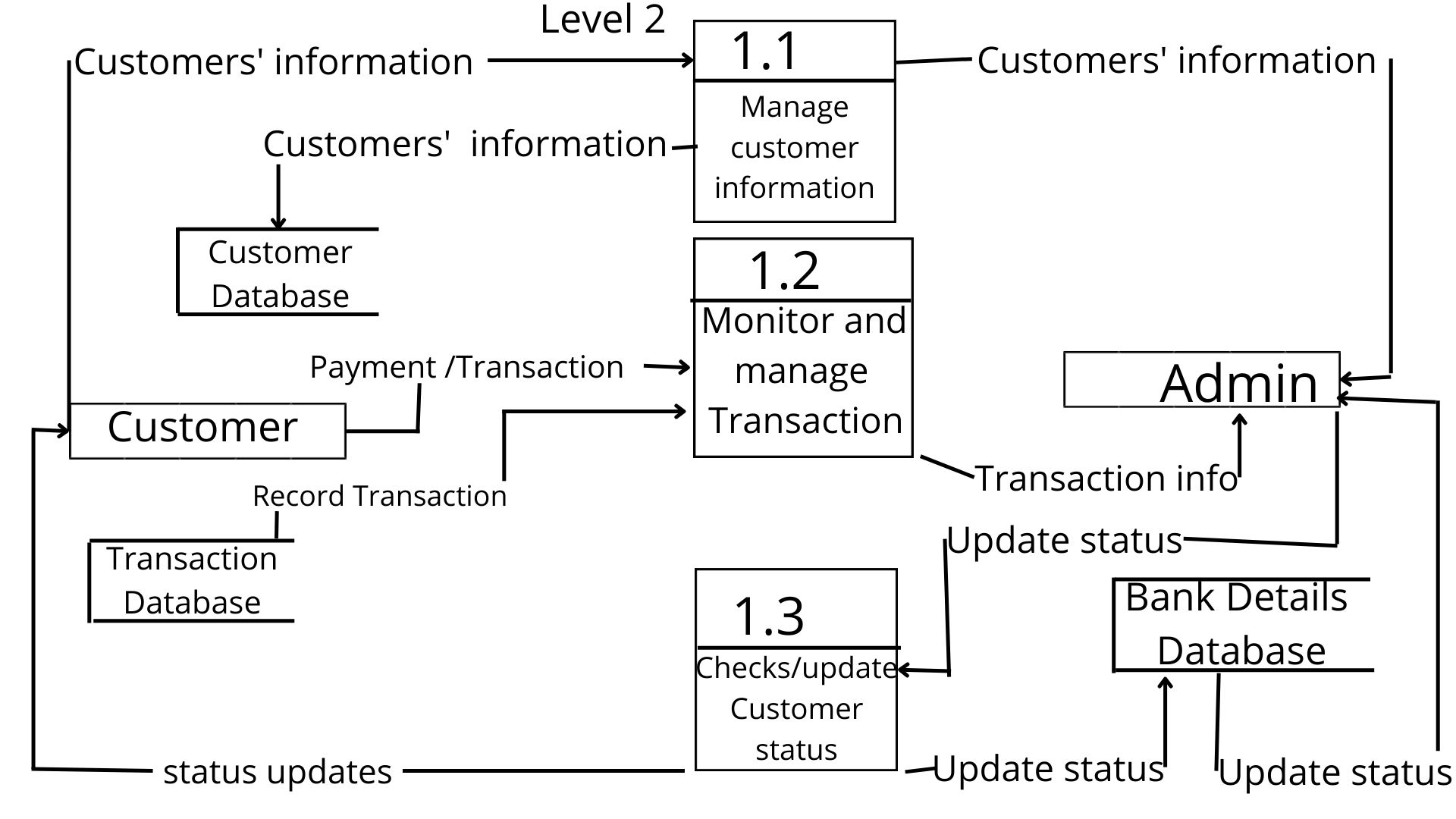
**Data Flow Diagram**

*Level 1 –*

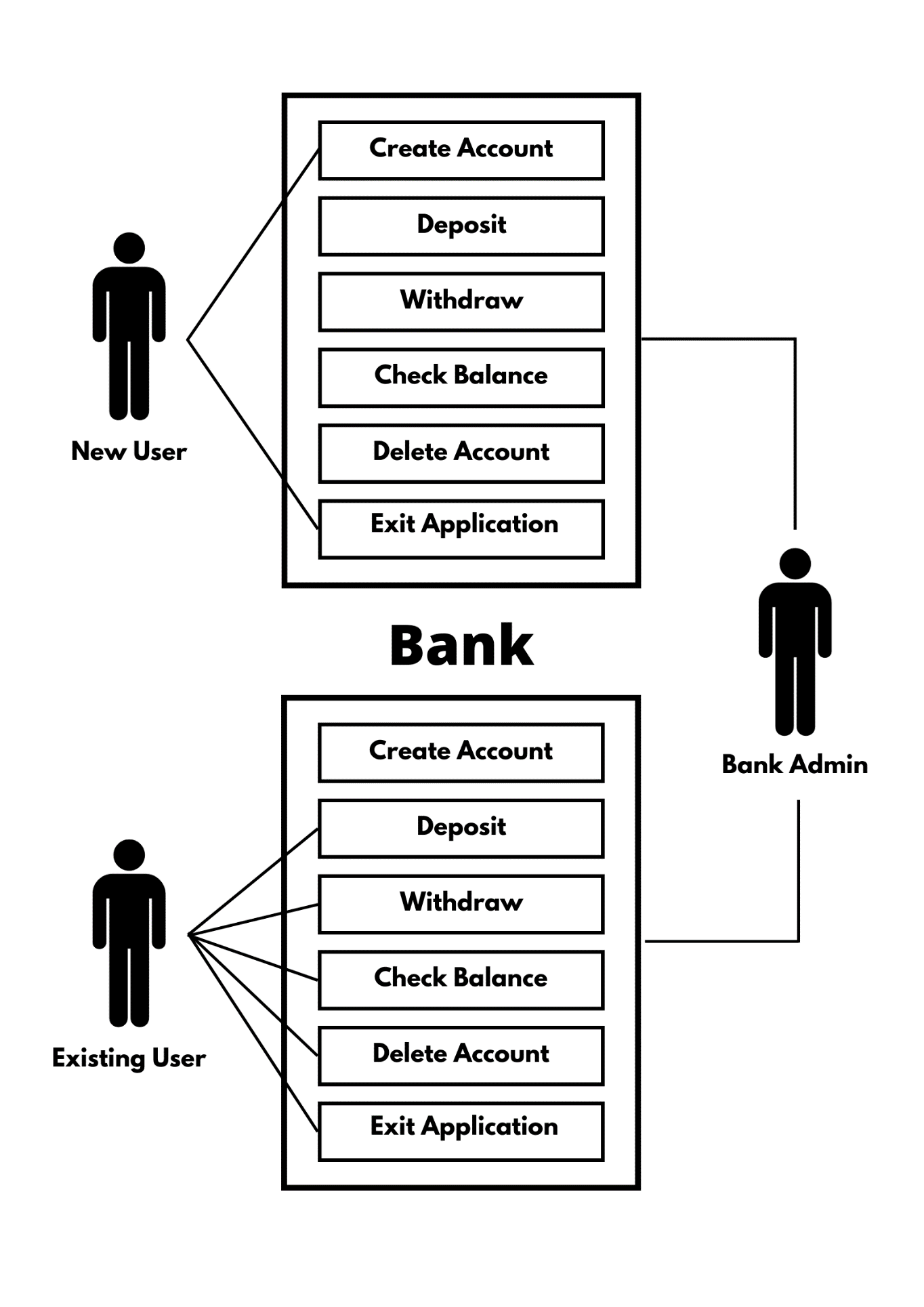
**

**Data Flow Diagram**

*Level 2 –*

****

**Use Case Diagram**

**Pseudo Codes :**

**Main File code**

package com.se.bankapplication;

import java.sql.\*;

import java.util.\*;

import java.lang.Thread;

import com.se.bankapplication.SQLite;

public class BankApplication {

public static void main(String args[]) {

SQLite.createTable();

Scanner sc = new Scanner(System.in);

int x = 0;

while (x == 0) {

System.out.println("WELCOME TO BANK APPlication\n");

System.out.println("Existing User? Press 1\n");

System.out.println("New User and want to create new account? Press 2\n");

System.out.println("Exit our application Press 3\n");

int opt = sc.nextInt();

if (opt == 1) {

int temp = 0;

int flag=0;

int acNum=0;

while (temp == 0) {

if(flag==0){

System.out.println("Enter your account number : ");

acNum = sc.nextInt();

Account currobj = SQLite.getDetails(acNum);

System.out.println("Enter your password :");

String pass=sc.next();

if (!pass.equals(currobj.password)){

System.out.println(pass.getClass().getSimpleName());

System.out.println("Wrong password " +pass + " " + currobj.password);

break;

}

flag=1;

}

//check acc

int ot;

System.out.println("\nChoose options as applicable, press assigned numeric values\n");

System.out.println("1: Check Balance");

System.out.println("2: Withdraw");

System.out.println("3: Deposit");

System.out.println("4: Delete Account");

System.out.println("5: Return to main menu");

ot = sc.nextInt();

switch (ot) {

case 1:

getDetails(acNum);

break;

case 2:

withdraw(acNum);

break;

case 3:

deposit(acNum);

break;

case 4:

deleteAccount(acNum);

temp = 1;

break;

case 5:

temp = 1;

break;

default:

System.out.println("Wrong choice, choose only from given options");

}

}

} else if (opt == 2) {

createAccount();

} else if (opt == 3) {

x = 1;

} else {

System.out.println("Wrong choice, choose again");

}

}

sc.close();

}

public static void withdraw(int acNum) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the amount to be withdrawn : ");

double amount = sc.nextDouble();

Account currobj = SQLite.getDetails(acNum);

if (currobj.balance >= amount) {

double balance = currobj.balance - amount;

currobj = SQLite.setBalance(acNum, balance);

System.out.println("Your remaining balance is " + currobj.balance);

} else {

System.out.println("Current balance is insufficient : " + currobj.balance);

}

}

public static void deposit(int acNum) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the amount to be deposited : ");

double depp = sc.nextDouble();

Account currobj = SQLite.getDetails(acNum);

double balance = currobj.balance + depp;

currobj = SQLite.setBalance(acNum, balance);

System.out.println("Rs." + depp + " successfully deposited to your account");

System.out.println("Your remaining balance is " + currobj.balance);

}

public static void getDetails(int acNum) {

Account currobj = SQLite.getDetails(acNum);

System.out.println("Name of Account holder: " + currobj.name);

System.out.println("Account Number of account holder: " + currobj.accountNumber);

System.out.println("Balance of account holder: " + currobj.balance);

}

static void createAccount() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the name : ");

String name = sc.nextLine();

System.out.print("Enter the age : ");

int age = sc.nextInt();

if(age<18){

System.out.println("Your age is less than the minimnum required age to open an account!!");

return;

}

System.out.println("Enter the address : ");

sc.next();

String address = sc.nextLine();

System.out.println("Enter the Password : ");

String occ = sc.next();

Account currobj = new Account(0, name, age, address, occ, 0, false);

int flag = SQLite.createAccount(currobj);

if (flag != -1) {

System.out.println("Account created successfully");

System.out.println("Your Account Number is : " + flag);

System.out.println("Please note down the account number for future transactions");

} else {

System.out.println("There was some error");

}

}

public static void deleteAccount(int acNum) {

Scanner sc = new Scanner(System.in);

System.out.println("Are you sure you want to delete your account ? Enter 'y' to continue. Enter 'n' to discard : ");

char st = sc.next().charAt(0);

if (st == 'y') {

System.out.println("Account deleted successfully");

boolean flag = SQLite.deleteAccount(acNum);

}

}

}

**SQLite Code:**

package com.se.bankapplication;

import java.sql.\*;

import com.se.bankapplication.Account;

import java.math.\*;

public class SQLite {

public static boolean createTable() {

Connection c = null;

Statement stmt = null;

boolean firstRun = false;

try {

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:bank.db");

stmt = c.createStatement();

String sql = "CREATE TABLE IF NOT EXISTS ACCOUNTS " +

"(ID INTEGER PRIMARY KEY AUTOINCREMENT," +

" NAME TEXT NOT NULL, " +

" AGE INT NOT NULL, " +

" PASSWORD TEXT NOT NULL, " +

" ADDRESS TEXT NOT NULL, " +

" BALANCE REAL)";

stmt.executeUpdate(sql);

stmt.close();

c.close();

updateId();

} catch ( Exception e ) {

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

return false;

}

return true;

}

public static void updateId() {

Connection c = null;

Statement stmt = null;

try {

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:bank.db");

c.setAutoCommit(false);

stmt = c.createStatement();

String sql = "UPDATE SQLITE\_SEQUENCE SET seq = 10000 WHERE name = 'ACCOUNTS'";

stmt.executeUpdate(sql);

c.commit();

stmt.close();

c.close();

} catch ( Exception e ) {

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

}

}

public static int createAccount( Account account) {

Connection c = null;

Statement stmt = null;

try {

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:bank.db");

c.setAutoCommit(false);

stmt = c.createStatement();

String sql = "INSERT INTO ACCOUNTS (NAME,AGE,PASSWORD,ADDRESS,BALANCE) " +

"VALUES (\"" + account.name + "\", " + String.valueOf(account.age) + ", \"" + account.password + "\", \"" + account.address + "\", 0.00 );";

stmt.executeUpdate(sql);

stmt.close();

c.commit();

c.close();

} catch ( Exception e ) {

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

return -1;

}

return countAll()-1;

}

public static Account getDetails( int accountNumber) {

Connection c = null;

Statement stmt = null;

Account ac = null;

try {

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:bank.db");

c.setAutoCommit(false);

stmt = c.createStatement();

ResultSet rs = stmt.executeQuery( "SELECT \* FROM ACCOUNTS WHERE id = " + String.valueOf(accountNumber) + ";" );

if (!rs.isBeforeFirst() ) {

ac = new Account(-1,"Error",-1,"Error","Error",-1,true);

return ac;

}

while ( rs.next() ) {

int id = rs.getInt("id");

String name = rs.getString("name");

int age = rs.getInt("age");

String address = rs.getString("address");

String password = rs.getString("password");

double balance = rs.getFloat("balance");

ac = new Account(id,name,age,address,password,balance,false);

}

rs.close();

stmt.close();

c.close();

} catch ( Exception e ) {

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

}

return ac;

}

public static Account setBalance(int accountNumber, double amount ) {

Connection c = null;

Statement stmt = null;

Account ac = null;

try {

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:bank.db");

c.setAutoCommit(false);

stmt = c.createStatement();

String sql = "UPDATE ACCOUNTS set BALANCE = " + String.valueOf(amount) + " where ID=" + String.valueOf(accountNumber) + ";";

stmt.executeUpdate(sql);

c.commit();

ResultSet rs = stmt.executeQuery( "SELECT \* FROM ACCOUNTS WHERE id = " + String.valueOf(accountNumber) + ";" );

if (!rs.isBeforeFirst() ) {

ac = new Account(-1,"Error",-1,"Error","Error",-1,true);

return ac;

}

while ( rs.next() ) {

int id = rs.getInt("id");

String name = rs.getString("name");

int age = rs.getInt("age");

String address = rs.getString("address");

String password = rs.getString("password");

double balance = rs.getFloat("balance");

ac = new Account(id,name,age,address,password,balance,false);

}

rs.close();

stmt.close();

c.close();

} catch ( Exception e ) {

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

return new Account(-1,"Error",-1,"Error","Error",-1,true);

}

return ac;

}

public static boolean deleteAccount( int accountNumber) {

Connection c = null;

Statement stmt = null;

try {

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:bank.db");

c.setAutoCommit(false);

stmt = c.createStatement();

String sql = "DELETE from ACCOUNTS where ID=" + String.valueOf(accountNumber) + ";";

stmt.executeUpdate(sql);

c.commit();

stmt.close();

c.close();

} catch ( Exception e ) {

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

return false;

}

return true;

}

public static void printAll( ) {

Connection c = null;

Statement stmt = null;

int id = 0;

try {

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:bank.db");

c.setAutoCommit(false);

stmt = c.createStatement();

ResultSet rs = stmt.executeQuery( "SELECT \* FROM ACCOUNTS;" );

while ( rs.next() ) {

id = rs.getInt("id");

String name = rs.getString("name");

int age = rs.getInt("age");

String address = rs.getString("address");

String password = rs.getString("password");

double balance = rs.getFloat("balance");

System.out.println( "ACCOUNT NUMBER = " + id );

System.out.println( "NAME = " + name );

System.out.println( "AGE = " + age );

System.out.println( "ADDRESS = " + address );

System.out.println( "PASSWORD = " + password );

System.out.println( "BALANCE = " + balance );

System.out.println();

}

rs.close();

stmt.close();

c.close();

} catch ( Exception e ) {

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

}

}

public static int countAll( ) {

Connection c = null;

Statement stmt = null;

int id = 0;

try {

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:bank.db");

c.setAutoCommit(false);

stmt = c.createStatement();

ResultSet rs = stmt.executeQuery( "SELECT \* FROM ACCOUNTS;" );

while ( rs.next() ) {

id = rs.getInt("id");

}

rs.close();

stmt.close();

c.close();

} catch ( Exception e ) {

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

return -1;

}

return id+1;

}

}

**Utility Class code :**

package com.se.bankapplication;

public class Account {

int accountNumber;

String name;

int age;

String password;

String address;

double balance;

boolean dataNotFound;

Account(int ac, String name, int age, String add, String pass, double bal, boolean dnf){

this.accountNumber = ac;

this.name = name;

this.age = age;

this.password = pass;

this.address = add;

this.balance = change(bal);

this.dataNotFound = dnf;

}

static double change(double value)

{

// Using the pow() method

value = value \* Math.pow(10, 2);

value = Math.floor(value);

value = value / Math.pow(10, 2);

return value;

}

}

**Project Risk**

Project Risks threaten the project plan that is, if project risk become real are likely project schedule will slip and that costs will increase. Project risk identify potential budgetary, schedule, personal, resource and requirement problems and their impact on a software project. Project complexity, size and degree of structural uncertainly where also defined as project risk factor.

I have distributed the possible risk the following categories and tried to find out the possible risk categories:

* Size-related risks: Risk related with the size of the project.
* Process definition: Risk associated with the degree to which I have defined my software installation process.
* Development environment risk: Risk associated with the availability and quality of the tools to be used to build the product.
* Technology risk: Technology availability and usability.
* Experience: Available experience in the concerned technology.

**Limitations**

* It doesn't covers special bank services like loan, money transfer, money order etc.
* Our application may not be very efficient, it may lack in terms of fast processing, slow service etc.
* Our application is currently having command line interface, it is still lacking to have a graphical user interface.
  + - Very few services are currently provided by our application, it will gradually inculcate further facilities.

**Conclusion**

*In this emerging world of computers all most all manual systems have been automated and computerized but maximum of them are so complex that a common user is unable to operate that software system. This website provides a computerized version of Bank management system which has a user-friendly interface and simple design, which ensures user-satisfaction. There is no chance of fault or miscalculation if the data is fed correctly. Use of the computers has solved many problems, which are faced while manual calculation.*

*Reference*

* www.w3schools.com
* www.roseindia.net
* www.dbforums.com
* www.ibm.com
* http://tomcat.apache.org/