ATM INTERFACE USING TCP/IP PROTOCOL

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE302: COMPUTER NETWORKS

Submitted by

S.SAKTHI PRABHA

124003263, B.Tech (CSE)

DECEMBER-2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR – 613 401

<u>Bonafide Certificate</u>

This is to certify that the report titled "**ATM Interface using TCP/IP Protocol**" submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **S.SAKTHI PRABHA (124003263, B.Tech(CSE)**) during the academic year 2022-23, in the School of Computing**.**

Project Based Work *Viva voc*e held on _______________

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENTS

First of all, I would like to thank God Almighty for his endless blessings.

I would like to express my sincere gratitude to **Dr S. Vaidyasubramaniam, Vice-Chancellor** for his encouragement during the span of my academic life at SASTRA Deemed University.

I would forever remain grateful and I would like to thank **Dr A.Umamakeswri, Dean, School of Computing and R. Chandramouli, Registrar** for their overwhelming support provided during my course span in SASTRA Deemed University.

I am extremely grateful to **Dr. Shankar Sriram, Associate Dean, School of Computing** for his constant support, motivation and academic help extended for the past three years of my life in School of Computing.

I would specially thank and express my gratitude to **Dr.Shankar Sriram, Associate Dean , School of Computing** for providing me an opportunity to do this project and for his guidance and support to successfully complete the project.

I also thank all the Teaching and Non-teaching faculty, and all other people who have directly or indirectly help me through their support, encouragement and all other assistance extended for completion of my project and for successful completion of all courses during my academic life at SASTRA Deemed University. Finally, I thank my parents and all others who help me acquire this interest in project and aided me in completing it within the deadline without much struggle.

List of Figures

List of Tables

ABBREVIATION

ATM	Automated Teller Machine
TCP/IP	Transmission Control Protocol/Internet Protocol
SMTP	Simple Mail Transfer Protocol
MTA	Mail Transfer Agent
UA	User Agent
PIN	Personal Identification Number
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
JAR	Java Archive
AES	Advanced Encryption Standard
JDBC	Java Database Connectivity

ABSTRACT

Automated Teller Machine(ATM) are electronic banking outlets that allow people to complete transactions without going to a branch of their bank. Some ATMs are simple cash deposits or for credits while others allow a variety of transactions such as check deposits, balance transfers and bill payments. To provide a secure and reliable service to the users, the bank and the ATM should be connected in a network so that the users can fetch money from any ATM of the respected bank. In this paper, we will discuss about implementation of TCP/IP protocol over ATM services to provide a reliable and secure financial transactions. TCP/IP protocol is the most reliable protocol that facilitates the exchange of messages between computers in a network.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

An automated teller machine (ATM) is an electronic telecommunication device that enables customers of financial institutions to perform financial transactions, such as cash withdrawals, cash deposits, fund transactions, balance inquiries, or account information inquiries, at anytime and anywhere without the need of interaction with associated staff. Customers are typically identified by inserting a ATM card into an ATM, with authentication being by the customer by entering a personal identification number (PIN), which must match the PIN stored in the chip on card, or in the issuing financial institution's database. For our project purpose we should manually type account number and PIN for verification. In the following report, ATM interface is implemented over TCP/IP protocol using Socket programming in Java to provide reliable service to customers is proposed. Customers can perform various financial transactions over a secured network.

CHAPTER 2

TCP/IP PROTOCOL

Computers exchange informations. Network is a collection of computers connected with each other. Routers, Switches are special form of computers. Each and every node of a network is called Network Element , it can be computer ,laptop, router, switch. Network refers to connecting nodes and communicating them with each other.

PROTOCOLS:

Protocols are common communication methodology used by computers and communication devices. The most common internet protocol is TCP/IP protocol. TCP protocol works in both wired and wireless network.

TCP/IP is a collection of the two main communication protocols such as TCP and IP. TCP stands for transmission control protocol and IP stands for internet protocol. TCP/IP uses a client/server model to send and receive data. TCP protocol is a connection-oriented protocol which provides reliable data transfer between two systems.

TCP makes sure that a physical connection is first established before start of data transfer. This physical connection is called channel. As the channel is established, the data flows in the same order as it is sent from the source.

THE SERVICE TCP PROVIDES TO AN APPICATION:

From an application program's point of view, the service offered by TCP has seven major features:

1. *Connection Orientation*. TCP provides connection-oriented service where an application first request a connection to a destination, and then use the connection to transfer data.

2. *Point-To-Point Communication*. Each TCP connection has exactly two endpoints.

3. *Complete reliability*. TCP guarantees that the data sent across a connection will be delivered exactly as sent, with no data missing or out of order.

4. *Full Duplex Communication*. A TCP connection allows data to flow in both direction, and allows both application program to send data at any time. TCP can buffer outgoing and incoming data in both directions, making it possible for an application to send data and then continue computation while the data is being transferred.

5. *Reliable Connection Startup*. TCP requires that when two applications create a connection, both must agree to the new connection. Duplicate packets used in previous connections will not appear to be valid responses or otherwise interfere with the new connection.

END TO END SERVICE:

The connections provided by TCP are called virtual connections . The TCP software modules on two machines exchange messages to achieve the illusion of a connection.

TCP uses IP to carry messages. Each TCP message is encapsulated in an IP datagram and sent across the internet. When the datagram arrives on the destination host, IP passes the contents to TCP. Although TCP uses IP to carry messages, IP does not read or interpret the messages. Thus, TCP treats IP as a packet communication system that connects hosts at two endpoints of a connection, and IP treats each TCP message as data to be transferred.

ACHIEVING RELIABILITY:

TCP uses a variety of techniques to achieve reliability. One of the most important techniques is retransmission. When TCP sends data, the sender compensates for packet loss by implementing a retransmission scheme. Both sides of a communication participate. When TCP receives data, it sends an acknowledgement back to the sender. Whenever it sends data, TCP starts a timer. If the timer expires before an acknowledgement arrives, the sender retransmits the data.

TCP's retransmission scheme is the key to its success because it handles communication across an arbitrary internet and allows multiple application programs to communicate concurrently.

Whenever it sends a message to which it expects a response, TCP records the time at which the message was sent. When a response arrives, TCP subtracts the time the message was sent from the current time to produce a new estimate of a round-trip delay for that connection. As it sends data packets and receive acknowledgements, TCP generates a sequence of round-trip estimates and uses a statistical function to produce a weighted average. In addition to a weighted average, TCP keeps an estimate of the variance, and uses a linear combination of the estimated mean and variance when computing the time at which retransmission is needed.

FLOW CONTROL AND WINDOW:

TCP uses a window mechanism to control the data flow. When connection is established, each end of a connection allocates a buffer to hold incoming data, and sends the size of the buffer to other end. As data arrives, the receiver sends acknowledgements, which is also specify the remaining buffer size. Buffer space available is called the window. A receiver sends a window advertisement with each acknowledgement.

THREE WAY HANDSHAKE:

To guarantee that connections are established or terminated reliably, TCP uses a three-way handshake in which three messages are exchanged. It is necessary and sufficient to ensure unambiguous agreement despite packet loss, duplication, and delay.

TCP uses the term synchronization segment(SYN segment) to describe messages in a 3-way handshake used to create a connection, and the term finished segment(FIN segment) to describe messages in a 3-way handshake used to close a connection.

TCP SEGMENT FORMAT:

TCP uses a single format for all messages, including messages that carry data, those that carry acknowledgements, and messages that are part of the 3-way handshake used to create and terminate a connection. TCP connection contains two streams of data, one flowing in each direction. If the applications at each end are sending data simultaneously, TCP can send a single segment that carries the acknowledgement for incoming data, a window advertisement that specifies the amount of additional buffer space available for incoming data, and outgoing data.

When a computer sends a segment, the acknowledgement number and window fields refer to incoming data. The acknowledgement number specifies the sequence of the data that is expected next, and the window specifies how much additional buffer space is available for data starting at the position given by the acknowledgement. The sequence number field refers to outgoing data. It gives the sequence number of the first octet of data being carried in the segment. The receiver uses the sequence number to reorder segments that arrive out of order and to compute an acknowledgement number. Destination port identifies which application program on the receiving computer should receive data, while source port identifies the application program that sends the data. Finally, the checksum field contains a checksum that covers the TCP segment header and the data.

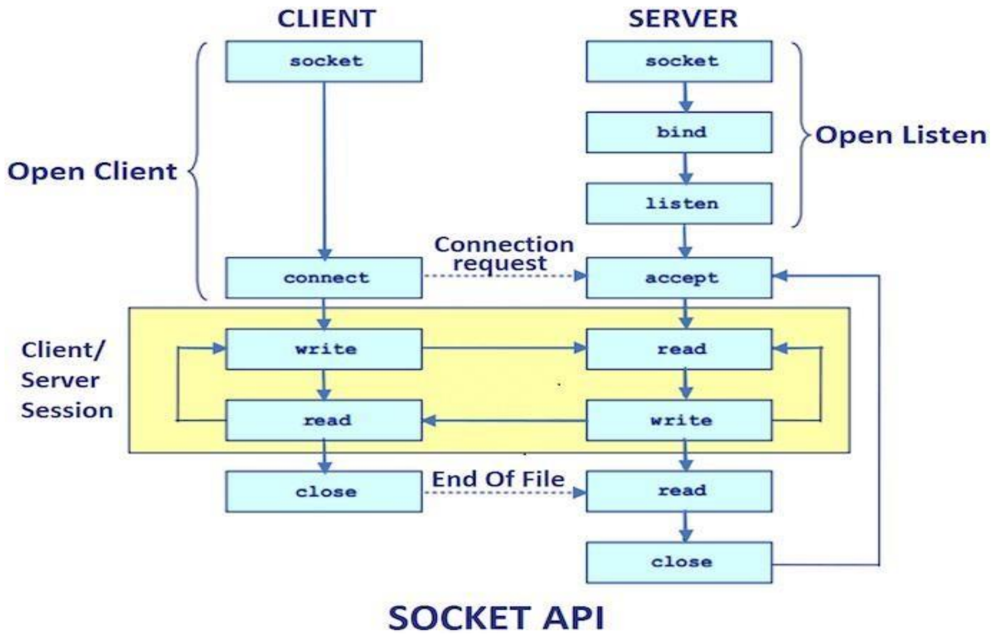
CHAPTER 3

SOCKET PROGRAMMING USING JAVA

Java Socket programming is used for communication between the applications runningon different JRE. Socket and ServerSocket classes are used for connection-oriented socket programming.

CLIENT SERVER COMPUTING:

When computers are connected in a network, In Client Server Server is capable of receiving requests from applications (called client) running on other computers and responds witha solution.



IP ADDRESS, PORT NUMBER SOCKET:

IP addresses are the fundamental method for computers to identify themselves on the Internet and many other computer networks. The IP Address of a computer is a quadruple where each member is a decimal from 0 to 255. Every computer has a single physical connection to a network. IP address is unique for a physical computer. However several applications can be running concurrently in the same machine. So, if a client wants to look up a particular application

running on a computer there must be a way of specification. So, port numbers are specified for each application. Port number is a 16bit number and hence decimally they are represented from 0to 65535. The port numbers 0 to 1023 are reserved for popular protocols such as HTTP, FTP and system services. We can use the other port numbers for user defined applications.

In a client-server applications, smooth data transfer takes place between the client program and the server program. This is a two-way communication. In the two-way communication, the communication channels are defined by Socket. A socket is generally defined to be one end pointof the two-way communication. A socket is specified by the IP Address and the Port Number of the application.

IMPORTANT METHODS FOR SERVER API:

Method	**Description**
1)public Socket accept()	returns the socket and establish a connection between server and client.
2)public synchronized void close()	Close the server socket

Table.1

IMPORTANT METHODS FOR CLIENT API:

Method	**Description**
public InputStream getInputStream()	returns the InputStream attached with this socket.
public OutputStream getOutputStream()	returns the OutputStream attached with this socket.
Public void close()	This will close the socket

Table.2

BUFFEREDREADER CLASS METHODS:

Java BufferedReader class is used to read the text from a character-based input stream. It can be used to read data line by line by readLine() method. It makes the performance fast.

Method	**Description**
int read()	It read the next byte of data from the input stream.
void close()	It closes the input stream and releases any of the system resources associated with the stream.

Table.3

Constructor	Description
BufferedInputStream(InputStream IS)	It creates the BufferedInputStream and saves it argument, the input stream IS, for later use.

Table.4

BUFFEREDWRITER CLASS METHODS:

Java BufferedWriter class is used to provide buffering for Writer instances. It makes the performance fast.

Method	Description
void write(int b)	It writes the specified byte to the buffered output stream.
void flush()	It flushes the buffered output stream.

Table.5

Constructor	Description
BufferedOutputStream(OutputStream os)	It creates the new buffered output stream which is used for writing the data to the specified output stream.

Table.6

CHAPTER 4

SIMPLE MAIL TRANSFER PROTOCOL

Whenever customer withdraws or deposits money in the ATM, they will receive a mail notification through their registered email-id with the bank. The required email-id of customer is fetched from bank's database and the system automatically generates email and sends to customer based on their transactions. For sending emails, we use SMTP.

SMTP is an application layer protocol. The client who wants to send the mail establishes a TCP connection to the SMTP server and then sends the mail across the connection. The SMTP server is always-on listening mode. Once it listens for a TCP connection from any client, the SMTP process creates a connection at port 25. After successfully establishing a TCP connection the client sends the mail .

In the SMTP model user deals with the user agent (UA), for example, Microsoft Outlook, etc. In order to exchange the mail using TCP, Mail Transfer Agent (MTA) is used. The user sending the mail doesn't have to deal with MTA as it is the responsibility of the system admin to set up a local MTA. The MTA maintains a queue of mails whicht it can schedule resends mail in case the receiver is not available. The MTA delivers the mail to the mailboxes .

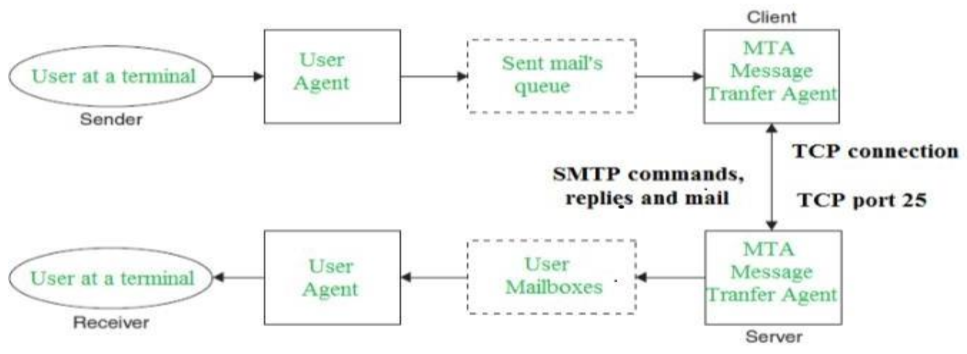


Fig.2-Schematic of SMTP

COMMUNICATION BETWEEN SENDER AND RECEIVER:

The sender's user agent prepares the message and sends it to the MTA. The MTA's responsibility is to transfer the mail across the network to the receiver's MTA. To send mails, a system must have a client MTA, and to receive mails, a system must have a server MTA.

SENDING EMAIL:

Mail is sent by a series of request and response messages between the client and the server. The message which is sent across consists of a header and a body. A null line is used to terminate the mail header and everything after the null line is considered as the body of the message, which is a sequence of ASCII characters. The message body contains the actual information read by the receipt.

RECEIVING EMAIL:

The user agent at the server-side checks the mailboxes at a particular time of intervals. If any information is received, it informs the user about the mail. When the user tries to read the mail it displays a list of emails with a short description of each mail in the mailbox. By selecting any of the mail users can view its contents on the terminal.

IMPLEMENTING SMTP IN JAVA:

To implement SMTP protocol in JAVA, we need an Java Mail API. We should download that API file and define that file in our project's class path. After that, we can use set of functions in order to send a mail to customers. For this purpose, we should have an email-id for bank and customers should provide their email-id at the time of account creation. Here we use Gmail server to send email to the customers.

STEPS TO SEND MAIL USING JAVA MAIL API:

1. Get session object – object that stores all the information of host like name, username, password etc.
2. Compose the message.
3. Send the Mail.

METHODS IN JAVA MAIL API:

METHOD	DESCRIPTION
public static session getDefaultInstance()	Returns a default session
Public void addRecipients()	Used to add the given address to the recipient type
public void setSubject()	Used to set subject header
public void setText()	Used to set the text as the message content

Table.7

CHAPTER 5
JAVA DATABASE CONNECTIVITY(JDBC)

In order to store large number of customers data, we use database to store them in a secure manner using SQLite database. Java Database Connectivity is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections. JDBC connections support creating and executing statements.

These may be update statements such as SQL's CREATE, INSERT, UPDATE AND DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

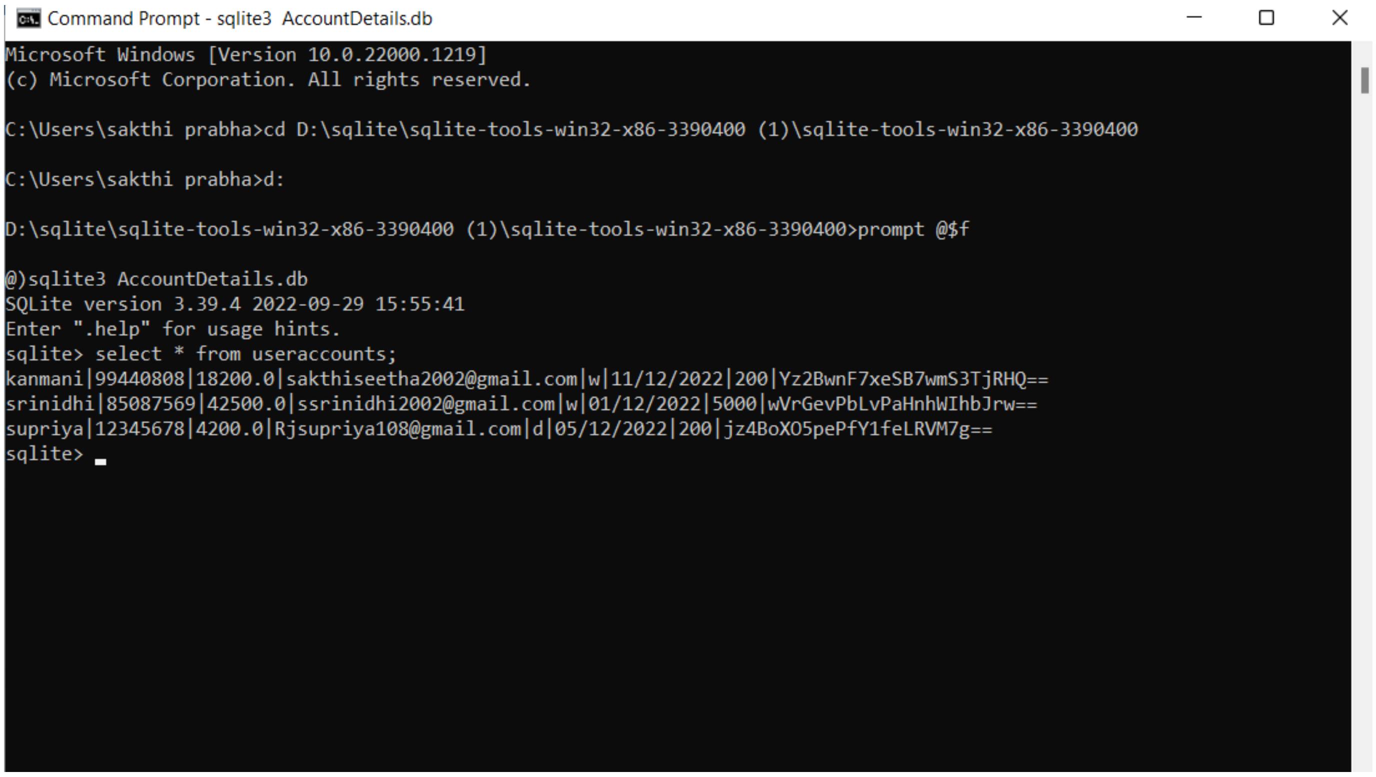
1. Statement- The statement is sent to the database server each and every time.
2. PreparedStatement- The statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.

Query statements return a JDBC row result set. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

For this project, we use SQLite3 database engine to store the data. The following steps is used to connect the database with our bank server:

1. Register the Driver class which is an executable JAR file. It’s path should be added in class path.
2. The getConnection() method of DriverManager class is used to establish connection with the database.
3. The createStatement() method of connection interface is used to create statement. The object of statement is responsible to execute queries with the database.
4. The executeQuery() method of statement interface is used to execute queries to the database. This method returns the object of Resultset that can be used to get required records of a table.
5. The close() method of Connection interface is used to close the connection.

For our project, we use SQLite3 Database to store the data. If we want to store large data we can use MySQL database and many more. To access our database, in command prompt we should navigate to the directory where database file is stored. After that we should enter the following commands to access our database:



```
Microsoft Windows [Version 10.0.22000.1219]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sakthi prabha>cd D:\sqlite\sqlite-tools-win32-x86-3390400 (1)\sqlite-tools-win32-x86-3390400

C:\Users\sakthi prabha>d:

D:\sqlite\sqlite-tools-win32-x86-3390400 (1)\sqlite-tools-win32-x86-3390400>prompt @$f

@)sqlite3 AccountDetails.db
SQLite version 3.39.4 2022-09-29 15:55:41
Enter ".help" for usage hints.
sqlite> select * from useraccounts;
kanmani|99440808|18200.0|sakthiseetha2002@gmail.com|w|11/12/2022|200|Yz2BwnF7xeSB7wmS3TjRHQ==
srinidhi|85087569|42500.0|ssrinidhi2002@gmail.com|w|01/12/2022|5000|wVrGevPbLvPaHnhWIhbJrw==
supriya|12345678|4200.0|Rjsupriya108@gmail.com|d|05/12/2022|200|jz4BoXO5pePfY1feLRVM7g==
sqlite>
```

Fig.3 Bank’s database

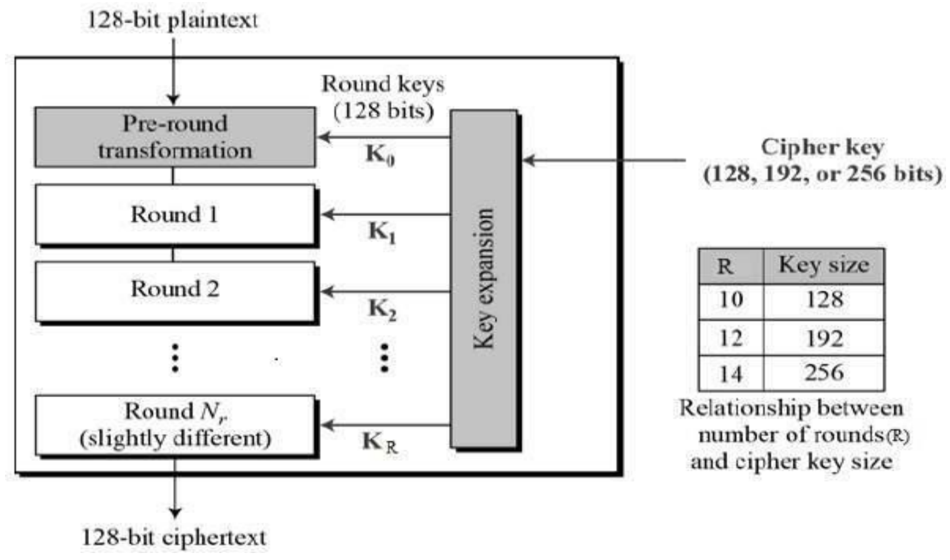
The database of our bank has a table named useraccounts. The useraccounts table contains number of columns such as customer’s name, account number, account balance, email-id, last transaction type, last transaction amount, last transaction date, pin in encrypted form. All of these data are stored in a secured manner in bank’s server. Only bank staffs can access this database.

CHAPTER 6

ADVANCED ENCRYPTION STANDARD(AES)

.

The more popular and widely adopted symmetric encryption algorithm is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. It uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.



R	Key size
10	128
12	192
14	256

Fig.4- Schematic of AES structure

ENCRYPTION PROCESS:

Each typical round of AES encryption comprises of four sub-processes:

1. Byte substitution- The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.
2. Shift rows- Each of the four rows of the matrix is shifted to the left.
3. Mix columns- Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.
4. Add round key- The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

BANK SERVER:

```
 import java.io.*;
import java.net.*;
import java.security.Key;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.Date;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.mail.*;
class BankServer extends Thread {

        ServerSocket serversocket=null;
        Socket socket=null;
        BufferedReader breader=null;
        BufferedWriter bwriter=null;
        Connection con=null;
        Statement statement=null;
        Scanner sc=null;
        HashMap<Integer,String> logininfo=new HashMap<Integer,String>();
        HashMap<Integer,String> nameinfo=new HashMap<Integer,String>();
        HashMap<Integer,Float> balanceinfo=new HashMap<Integer,Float>();
```

```
    HashMap<Integer,String> emailinfo=new HashMap<Integer,String>();
    HashMap<Integer,String> ltrtinfo=new HashMap<Integer,String>();
    HashMap<Integer,String> ltramtinfo=new HashMap<Integer,String>();
    HashMap<Integer,String> dateinfo=new HashMap<Integer,String>();
    Session newSession=null;
    MimeMessage mimeMessage = null;
    SimpleDateFormat day=new SimpleDateFormat("dd/MM/yyyy");
    SimpleDateFormat time=new SimpleDateFormat("HH:mm");
    Date date=new Date();
    String skey="AHGDU284992HFSHD";
    Key key=new SecretKeySpec(skey.getBytes(),"AES");

    public BankServer(Socket s,BufferedReader br,BufferedWriter bw)
    {
        this.socket=s;
        this.breader=br;
        this.bwriter=bw;
    }

@Override
    public void run()
    {

    int amt ;
    int uaccno;
    String upin="",upin2;
    float newbalance;
```

```
String mode,s1;
managedb();
uaccno=Integer.parseInt(readfromclient());
System.out.print("\nAccount number:"+uaccno);
upin2=readfromclient();
try {
                upin=encrypt(upin2,key);
        } catch (Exception e1) {
                e1.printStackTrace();
        }
if(logininfo.containsKey(uaccno))
        {
        if(logininfo.get(uaccno).equalsIgnoreCase(upin))
        {
        sendtoclient("valid");
        String name=nameinfo.get(uaccno);
                sendtoclient(name);
        while(true) {
                mode=readfromclient();
                setupServerProperties();
                if (mode.equalsIgnoreCase(Character.toString('w')))
                {
                        amt=Integer.parseInt(readfromclient());
                        Float balance=balanceinfo.get(uaccno);
                        if(balance-amt>0)
                        {
                                newbalance=balance-amt;
```

```
				s1="UPDATE useraccounts SET Balance="+newbalance+" WHERE AccountNumber="+uaccno;

				String s2="UPDATE useraccounts SET ltransactiondate='"+day.format(date)+"',ltransactiontype='w',ltransactionamt='"+Integer.toString(amt)+"' WHERE AccountNumber="+uaccno;

				try

				{

				statement=con.createStatement();

				statement.executeUpdate(s1);

				statement.executeUpdate(s2);

				}

				catch(Exception e)

				{

					e.printStackTrace();

				}

				sendtoclient(Float.toString(newbalance));

				draftEmail(uaccno,amt,mode,emailinfo.get(uaccno));

				try {

					sendEmail();

				} catch (MessagingException e) {

					e.printStackTrace();

				}

				break;

			}

			else

			{

					sendtoclient("Invalid Balance");

			}
```

```
                    }
                    else if(mode.equalsIgnoreCase(Character.toString('d')))
                    {
                            amt=Integer.parseInt(readfromclient());
                            Float balance=balanceinfo.get(uaccno);
                            newbalance=balance+amt;
                            s1="UPDATE useraccounts SET Balance="+newbalance+" WHERE
AccountNumber="+uaccno;
                            String s2="UPDATE useraccounts SET
ltransactiondate='"+day.format(date)+"',ltransactiontype='d',ltransactionamt='"+Integer.toString(amt)+"
' WHERE AccountNumber="+uaccno;
                            try
                            {
                            statement=con.createStatement();
                            statement.executeUpdate(s1);
                            statement.executeUpdate(s2);
                            }
                            catch(Exception e)
                            {
                                    e.printStackTrace();
                            }
                            sendtoclient(Float.toString(newbalance));
                            draftEmail(uaccno,amt,mode,emailinfo.get(uaccno));
                            try {
                                    sendEmail();
                            } catch (MessagingException e) {
                                    e.printStackTrace();
                            }
```

```
                        break;


                }
                else if(mode.equalsIgnoreCase(Character.toString('b')))
                {


                        Float balance=balanceinfo.get(uaccno);
                        sendtoclient(Float.toString(balance));
                }
                else if(mode.equalsIgnoreCase(Character.toString('t')))
                {
                        String tr=ltrtinfo.get(uaccno);
                        String tramt=ltramtinfo.get(uaccno);
                        String date2=dateinfo.get(uaccno);
                   sendtoclient(tr);
                   sendtoclient(tramt);
                   sendtoclient(date2);
                }
        }
        }
                else
                {
                        System.out.print("\nInvalid Account Number or Pin....");
                        sendtoclient("Invalid");
                }
        }
        else
```

```
            {
                        System.out.print("\nInvalid Account Number or Pin....!");
                        sendtoclient("Invalid");
            }
    }

    private void sendEmail() throws MessagingException {
            String fromUser = "sakthiseetha2002@gmail.com";
            String fromUserPassword = "bgypzxhdytudrlmo";
            String emailHost = "smtp.gmail.com";
            Transport transport = newSession.getTransport("smtp");
            transport.connect(emailHost, fromUser, fromUserPassword);
            transport.sendMessage(mimeMessage, mimeMessage.getAllRecipients());
            transport.close();
            System.out.println("\nEmail has been successfully sent!!!");
    }
  private MimeMessage draftEmail(int accno,int amt,String mode,String emailid)  {
            String emailReceipients =emailid;
            String emailSubject = "KYC Bank \n\nAmount Transaction";
            String emailBody="";
            String emailBody1,emailBody2;

            if(mode.equals("w"))
            {
                        emailBody1 = "Your AccountNumber "+accno+" is withdrawed for Rs."+amt+" from
Thanjavur branch on "+day.format(date)+" at "+time.format(date);
                        emailBody2="\n\n\n\n\nThis is a system generated mail,do not reply to
```

```
 this mail.";
                        emailBody=emailBody1+emailBody2;
                }
                else
                {
                        emailBody1= "Your AccountNumber "+accno+" is deposited for
Rs."+amt+" to Thanjavur branch on "+day.format(date)+" at "+time.format(date);
                        emailBody2="\n\n\n\n\nThis is a system generated mail,do not reply
 to this mail.";
                        emailBody=emailBody1+emailBody2;
                }
                mimeMessage = new MimeMessage(newSession);
                try {
                mimeMessage.addRecipient(Message.RecipientType.TO, new InternetAddress(emailReceipients));
                mimeMessage.setSubject(emailSubject);
                mimeMessage.setText(emailBody);
                 return mimeMessage;
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
                return null;
        }


        private void setupServerProperties() {
                Properties properties = System.getProperties();
```

```
            properties.put("mail.smtp.port", "587");

            properties.put("mail.smtp.auth", "true");

            properties.put("mail.smtp.starttls.enable", "true");

            newSession = Session.getDefaultInstance(properties,null);

    }




     public void sendtoclient(String data)

        {

            try {

                    //data=encrypt(data,key);

                            bwriter.write(data);

                            bwriter.newLine();

                            bwriter.flush();

                    } catch (Exception e) {


                            System.out.print("\nError in sending message......");

                            e.printStackTrace();

                            CloseEverything(socket,breader,bwriter);

                    }


        }

     public String readfromclient()

        {

            String recmsg;

            try {
                    recmsg=breader.readLine();

                    System.out.println(recmsg);

                    return recmsg;

                }

        catch (Exception e) {

                    System.out.print("\nError in reading message...");

                    e.printStackTrace();

                    CloseEverything(socket,breader,bwriter);

                    return null;


                }

    }


    public void managedb()

    {

        String jdbcurl="jdbc:sqlite://D:\\sqlite\\sqlite-tools-win32-x86-3390400 (1)\\sqlite-tools-
win32-x86-3390400\\AccountDetails.db";

                try{

                    con =DriverManager.getConnection(jdbcurl);

                    String sql="SELECT * from useraccounts";

                    statement=con.createStatement();

                    ResultSet result=statement.executeQuery(sql);

                    while(result.next())

                    {

                        String pin=result.getString("Pin");

                        int accountno=result.getInt("AccountNumber");

                        String name=result.getString("name");
```

```
                                        Float balance=result.getFloat("Balance");
                                        String email=result.getString("emailid");
                                        String ltrt=result.getString("ltransactiontype");
                                        String ltramt=result.getString("ltransactionamt");
                                        String date1=result.getString("ltransactiondate");
                                        logininfo.put(accountno,pin);
                                        nameinfo.put(accountno,name);
                                        balanceinfo.put(accountno,balance);
                                        emailinfo.put(accountno,email);
                                        ltrtinfo.put(accountno,ltrt);
                                        ltramtinfo.put(accountno, ltramt);
                                        dateinfo.put(accountno, date1);
                                  }
                            }
                            catch(Exception e) {
                            System.out.print("\nError in fetching account number and PIN....");
                            e.printStackTrace();
                            }
  }
public void CloseEverything(Socket s,BufferedReader br,BufferedWriter bw)
  {
        try {
        if(s!=null)
              s.close();
        if(br!=null)
              br.close();
        if(bw!=null)
```

```
                        bw.close();
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }

        private String encrypt(String ct,Key key) throws Exception
                {
                Cipher c=Cipher.getInstance("AES");
                c.init(Cipher.ENCRYPT_MODE,key);
                byte[] encval=c.doFinal(ct.getBytes());
                String encstring=Base64.getEncoder().encodeToString(encval);
                return encstring;
                }




}

public class Bank_server
{
        public static  void main(String args[]) throws IOException
        {
                Socket socket;
                ServerSocket ss=null;
                BufferedReader breader=null;
                BufferedWriter bwriter=null;
                ss=new ServerSocket(2754);
                System.out.print("\nWaiting for client.....");
                while(true)
                {
                        try {
                                socket=ss.accept();
                                System.out.print("\nConnection has been  Established");
                                System.out.print("\nConnected to "+socket.getInetAddress().getHostName());
                                breader=new BufferedReader(new InputStreamReader(socket.getInputStream()));
                                bwriter=new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
                                BankServer b1=new BankServer(socket,breader,bwriter);
                                b1.start();
                        }
                        catch(IOException e)
                        {
                                System.out.print("\nError in creating thread....");
                                e.printStackTrace();


                        }
                }
        }
}
```

ATM CLIENT:

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.*;
import java.net.*;
import java.security.Key;
import java.util.Base64;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

public class Atm_client implements ActionListener
{
        private static Socket socket=null;
        private static BufferedReader breader=null;
        private static BufferedWriter bwriter=null;
        static String name,balance;
        static String accno;

        JFrame frame=new JFrame();
        JButton cd=new JButton("Cash deposit");
        JButton cw=new JButton("Cash withdrawal");
        JButton cb=new JButton("View Balance");
        JButton tr=new JButton("Last Transaction");
        JLabel welcomelabel=new JLabel();
        static JLabel welcomelabel1=new JLabel();
        static JLabel welcomelabel2=new JLabel();
        JLabel balancelabel=new JLabel();
        JLabel balancelabel2=new JLabel();
        JLabel balancelabel3=new JLabel();
        JLabel messagelabel=new JLabel();
        JTextField amt=new JTextField();
        JLabel amtlabel=new JLabel();
        static String skey="AHGDU284992HFSHD";
        static Key key=new SecretKeySpec(skey.getBytes(),"AES");


        public static void getConnection()
        {
```

```
		try
		{
			socket=new Socket("localhost",2754);
			System.out.print("\nConnected to "+socket.getInetAddress().getHostName());
			breader=new BufferedReader(new InputStreamReader(socket.getInputStream()));
			bwriter=new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
		}
		catch(Exception e)
		{
			System.out.print("\nError in Client.....");
			e.printStackTrace();
			closeEverything(socket,breader,bwriter);
		}
	}


	public Atm_client(String userid,String name)
	{
		cd.setBounds(325,350,150,25);
		cw.setBounds(125,350,150,25);
		cb.setBounds(125,400,150,25);
		tr.setBounds(325,400,150,25);
		balancelabel.setBounds(75,200,400,25);
		balancelabel.setFont(new Font(null,Font.BOLD,20));
		balancelabel2.setBounds(75,225,300,25);
		balancelabel2.setFont(new Font(null,Font.BOLD,18));
		balancelabel3.setBounds(75,250,300,25);
		balancelabel3.setFont(new Font(null,Font.BOLD,18));
		messagelabel.setBounds(150,500,300,35);
		messagelabel.setFont(new Font(null,Font.BOLD,25));

		amtlabel.setText("Amount:");
		amt.setBounds(155,150,150,25);
		amtlabel.setBounds(75,150,50,25);
		welcomelabel.setBounds(75,50,400,25);
		welcomelabel1.setBounds(75,90,300,25);
		welcomelabel2.setBounds(75,115,300,25);
		welcomelabel.setText("Welcome to our KYC ATM");
		welcomelabel.setFont(new Font(null,Font.BOLD,30));
		welcomelabel1.setFont(new Font(null,Font.BOLD,15));
		welcomelabel2.setFont(new Font(null,Font.BOLD,15));
		cd.setFocusable(false);
		cw.setFocusable(false);
		cb.setFocusable(false);
		tr.setFocusable(false);

		frame.add(amt);
		frame.add(amtlabel);
		frame.add(cd);
		frame.add(cw);
		frame.add(cb);
		frame.add(tr);
		frame.add(welcomelabel);
		frame.add(welcomelabel1);
		frame.add(balancelabel);
		frame.add(balancelabel2);
		frame.add(balancelabel3);
		frame.add(messagelabel);
		frame.setTitle("Transaction Window");
		frame.add(welcomelabel2);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setSize(650,650);
		frame.setLayout(null);
		frame.setVisible(true);
		frame.setResizable(false);
		cd.addActionListener(this);
		cw.addActionListener(this);
		tr.addActionListener(this);
		cb.addActionListener(this);
		welcomelabel1.setText("Name:"+name);
		welcomelabel2.setText("Acccount Number:"+userid);

	}
	public Atm_client()
	{

	}

	public void actionPerformed(ActionEvent e)
	{
		if(e.getSource()==cd)
		{
```

```
                sendtoserver(Character.toString('d'));
                String amount = amt.getText();
                sendtoserver(amount);
                balance=receivefromserver();
                balancelabel.setText("Balance:"+balance);
                messagelabel.setText("Transaction Successful!");
                JOptionPane.showMessageDialog(frame,"Cash deposited sucessfully");
                frame.dispose();
                new endpage();
            }
            if(e.getSource()==cw)
            {
                sendtoserver(Character.toString('w'));
                String amount = amt.getText();
                sendtoserver(amount);
                balance=receivefromserver();
                if(balance.equals("Invalid Balance"))
                {
                    messagelabel.setText("Transaction Failed!");
                    JOptionPane.showMessageDialog(frame,"Cash not
dispersed","Alert",JOptionPane.WARNING_MESSAGE);

                }
                else
                {
                balancelabel.setText("Balance:"+balance);
                messagelabel.setText("Transaction Successful!");
                JOptionPane.showMessageDialog(frame,"Cash withdrawed sucessfully");
                frame.dispose();
                new endpage();
                }
            }
            if(e.getSource()==cb)
            {
                sendtoserver(Character.toString('b'));
                balance=receivefromserver();
                balancelabel.setText("Balance:"+balance);

            }
            if(e.getSource()==tr)
            {
                    sendtoserver(Character.toString('t'));
                    String trt=receivefromserver();
                    String tramt=receivefromserver();
                    String date=receivefromserver();
                    if(trt.equals(Character.toString('w')))
                    {
                    balancelabel.setText("Last Transaction type:Withdrawl");
                    balancelabel2.setText("Last Transaction Amount:"+tramt);
                    balancelabel3.setText("Last Transaction Date:"+date);
                    }
                    else
                    {
                            balancelabel.setText("Last Transaction type:Deposit");
                            balancelabel2.setText("Last Transaction Amount:"+tramt);
                            balancelabel3.setText("Last Transaction Date:"+date);
                    }
                    }

    }
    public static String getdata(String uaccno,String pin) throws Exception
    {

            sendtoserver(uaccno);
            sendtoserver(pin);
            String p=receivefromserver();
            if(p.equals("valid"))
            {
                    return "valid";
            }
            else
            {
                    closeEverything(socket,breader,bwriter);
                    return "0";
            }
    }

    private static void sendtoserver(String data)
    {
            try {
                    data=(data);
                    System.out.println("......"+data);
```

```
                bwriter.write(data);
                bwriter.newLine();
                bwriter.flush();
        } catch (Exception e) {

                System.out.print("\nError in sending message......");
                e.printStackTrace();
                closeEverything(socket,breader,bwriter);
        }
}

public static String receivefromserver()
{
        String recmsg;
try {
                recmsg=breader.readLine();
                System.out.println("????"+recmsg);
                recmsg=(recmsg);

                return recmsg;
        } catch (Exception e) {
                System.out.print("\nError in reading message...");
                e.printStackTrace();
                closeEverything(socket,breader,bwriter);
                return null;
        }
}

private static void closeEverything(Socket s,BufferedReader br,BufferedWriter bw)
{
        try {
        if(s!=null)
                s.close();
        if(br!=null)
                br.close();
        if(bw!=null)
                bw.close();
        }
        catch(Exception e)
        {
                e.printStackTrace();
                }
        }
        private static String decrypt(String pt,Key key) throws Exception
        {
        Cipher c=Cipher.getInstance("AES");
        c.init(Cipher.DECRYPT_MODE,key);
        byte[] dec=Base64.getDecoder().decode(pt);
        byte[] decval=c.doFinal(dec);
        String decstring =new String(decval);
        return decstring;
        }

        public static String encrypt(String ct,Key key) throws Exception
        {
        Cipher c=Cipher.getInstance("AES");
        c.init(Cipher.ENCRYPT_MODE,key);
        byte[] encval=c.doFinal(ct.getBytes());
        String encstring=Base64.getEncoder().encodeToString(encval);
        return encstring;
        }
}

class endpage
{
        JFrame frame1=new JFrame();
        JLabel label=new JLabel();
        JLabel label2=new JLabel();
        JLabel label3=new JLabel();

        public endpage()
        {
                label.setText("Thank you for using KYC ATM services");
                label.setFont(new Font(null,Font.BOLD,27));
                label.setBounds(75,50,500,35);
                label2.setBounds(75,500,500,35);
                label2.setFont(new Font(null,Font.BOLD,18));
                label2.setText("For more details contact Queries@KYCbank.com");
                label3.setBounds(150,550,300,35);
                label3.setFont(new Font(null,Font.BOLD,18));
                label3.setText("CONTACT NUMBER:9651123457");
                frame1.setTitle("KYC Bank");
        frame1.add(label);
        frame1.add(label2);
        frame1.add(label3);
        frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame1.setSize(650,650);
        frame1.setLayout(null);
        frame1.setVisible(true);
        frame1.setResizable(false);

    }
```

LOGIN PAGE

```
import java.awt.Font;
import java.awt.event.*;
import javax.swing.*;

public class Loginpage implements ActionListener
{
        JFrame frame=new JFrame();
        JButton loginbutton=new JButton("Login");
        JButton resetbutton=new JButton("Reset");
        JTextField userlogin=new JTextField();
        JPasswordField password=new JPasswordField();
        JLabel useridlabel=new JLabel("User Account Number:");
        JLabel userpasslabel=new JLabel("User PIN:");
        JLabel messagelabel=new JLabel();
        JLabel welcomelabel=new JLabel();
        String userid="",name;
        static String mode="";
        Atm_client a1;

        public Loginpage()
        {
                useridlabel.setBounds(50,100,150,25);
                userpasslabel.setBounds(50,150,150,25);
                messagelabel.setBounds(125,250,250,35);
                messagelabel.setFont(new Font(null,Font.ITALIC,25));
                welcomelabel.setBounds(75,50,400,25);
                welcomelabel.setFont(new Font(null,Font.BOLD,20));
                welcomelabel.setText("WELCOME TO KYC BANK ATM");
                userlogin.setBounds(200,100,200,25);
                password.setBounds(200,150,200,25);
        loginbutton.setBounds(125,200,100,25);
        loginbutton.setFocusable(false);
        loginbutton.addActionListener(this);
        resetbutton.setBounds(225,200,100,25);
        resetbutton.setFocusable(false);
        resetbutton.addActionListener(this);

        frame.add(useridlabel);
        frame.add(userpasslabel);
        frame.setTitle("KYC ATM Login");
        frame.add(userlogin);
        frame.add(password);
        frame.add(messagelabel);
        frame.add(welcomelabel);
        frame.add(loginbutton);
        frame.add(resetbutton);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(650,650);
        frame.setLayout(null);
        frame.setVisible(true);
        frame.setResizable(false);
}

public void actionPerformed(ActionEvent e) {
        if(e.getSource()==resetbutton)
        {
                userlogin.setText("");
                password.setText("");
        }
        if(e.getSource()==loginbutton)
        {
                userid=userlogin.getText();
```

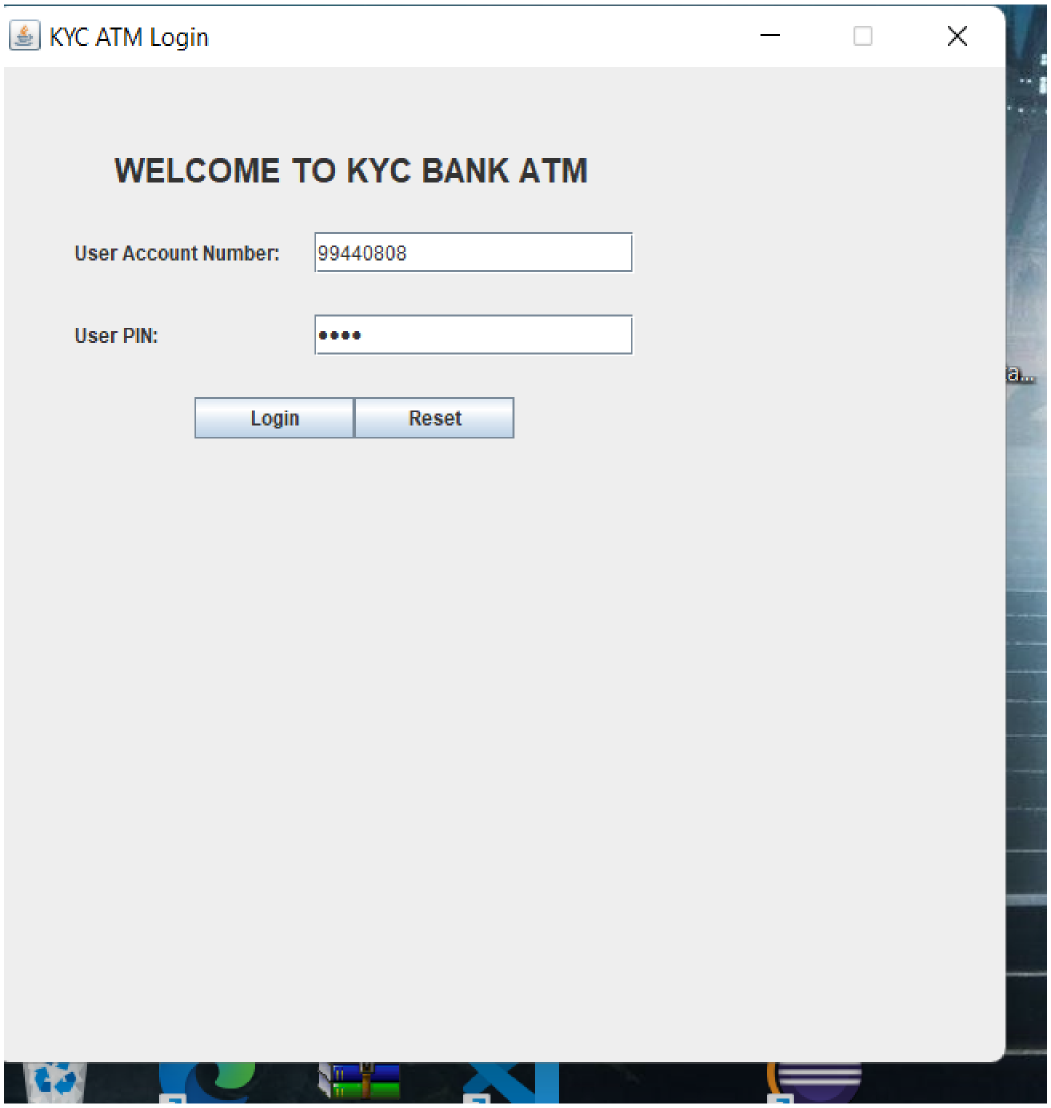
```
                        String password1=String.valueOf(password.getPassword());
                        try
                        {
                                mode=Atm_client.getdata(userid,password1);
                        } catch (Exception e1)
                        {
                                e1.printStackTrace();
                        }
                        if(mode.equals("valid"))
                        {
                                JOptionPane.showMessageDialog(frame,"SUCCESSFUL LOG IN!");
                                frame.dispose();
                                name=Atm_client.receivefromserver();
                                new Atm_client(userid,name);
                        }
                        else
                        JOptionPane.showMessageDialog(frame,"Invalid Account number or
PIN","Alert",JOptionPane.WARNING_MESSAGE);
                }
        }

        public static void main(String args[])
        {
                Atm_client.getConnection();
                new Loginpage();
        }
```

CHAPTER 8

SNAPSHOTS

After client connectes with the server the client needs to put the necessary details :

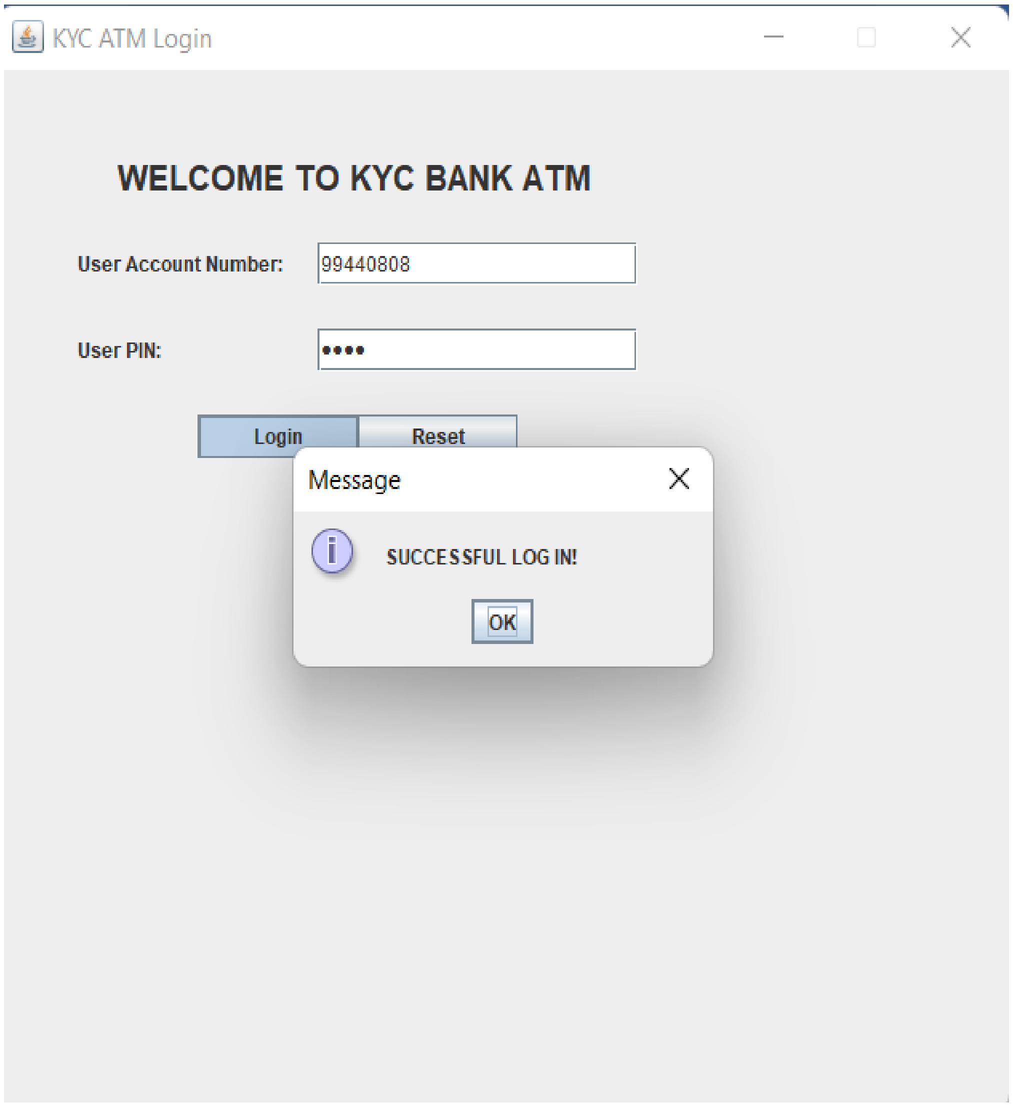
User Account Number and User PIN.



User Account Number: 99440808

User PIN: ••••

Client’s account number and pin the pin will be verified by crosschecking the database. If the Account number and pin are correct then it shows successful log in . After get verified by the bank server, the popup message will appear and redirected to the transaction window:


KYC ATM Login

WELCOME TO KYC BANK ATM

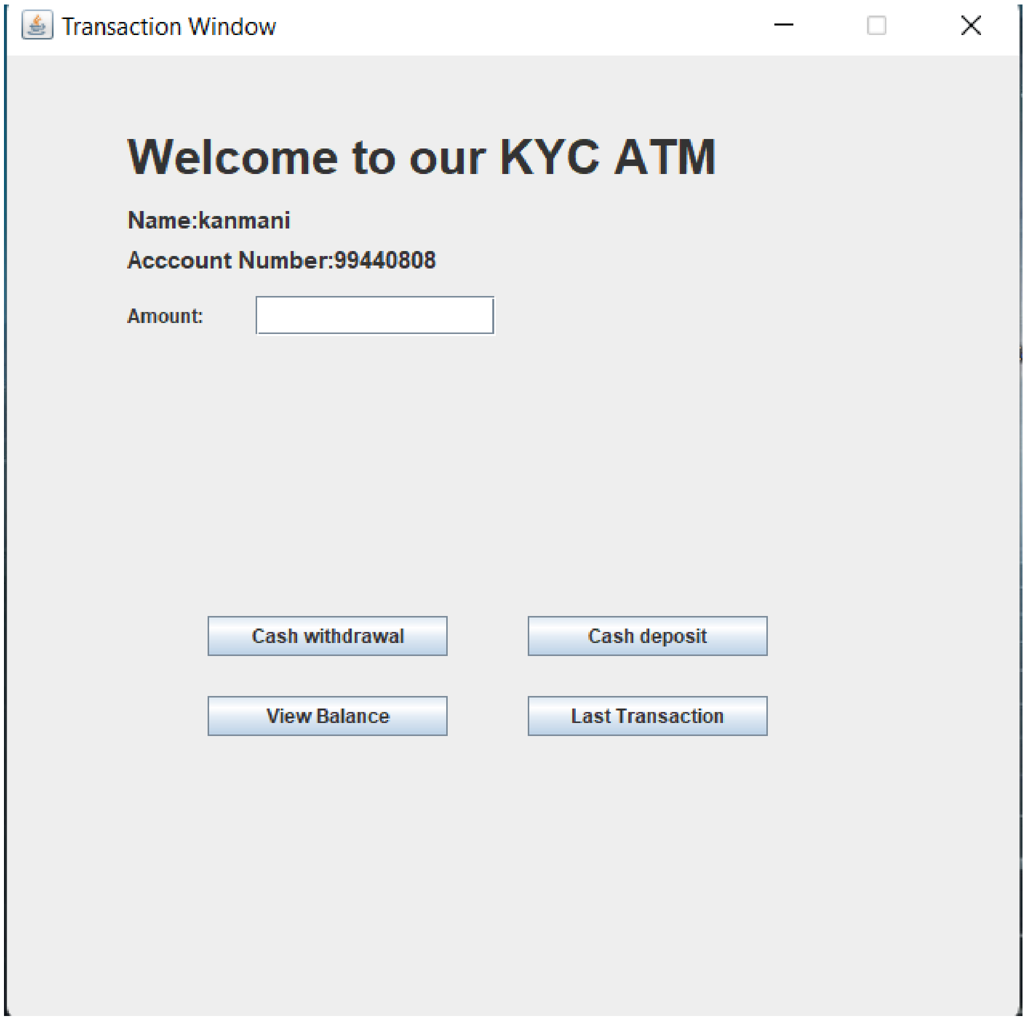
User Account Number: 99440808

User PIN: ••••

Message

SUCCESSFUL LOG IN!

After successfully verified by the bank server, we will be redirected to the transaction window:



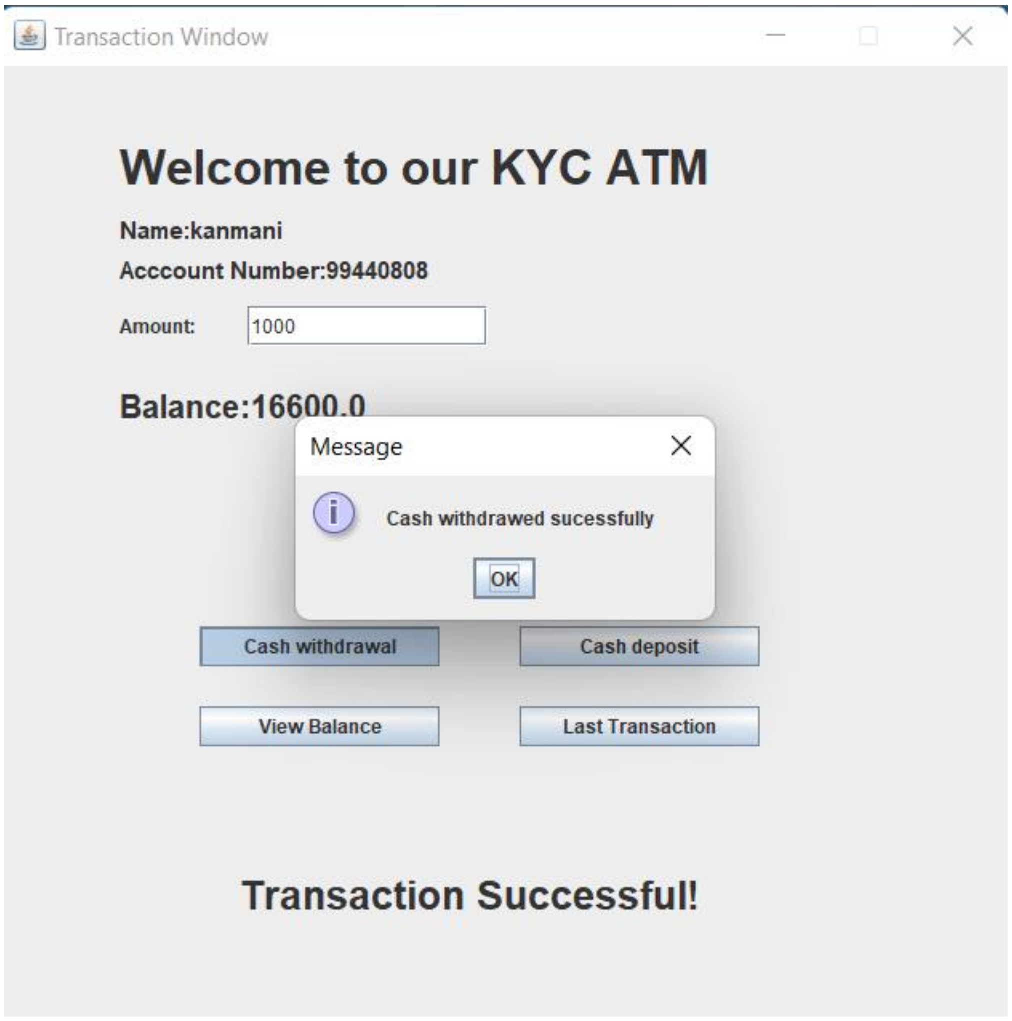
Welcome to our KYC ATM

Name:kanmani

Acccount Number:99440808

Amount:

In transaction window, enter the amount and click the cash withdraw button to withdraw the cash from the ATM and popup message will appear when transaction is successful:



Transaction Window

Welcome to our KYC ATM

Name:kanmani

Acccount Number:99440808

Amount: 1000

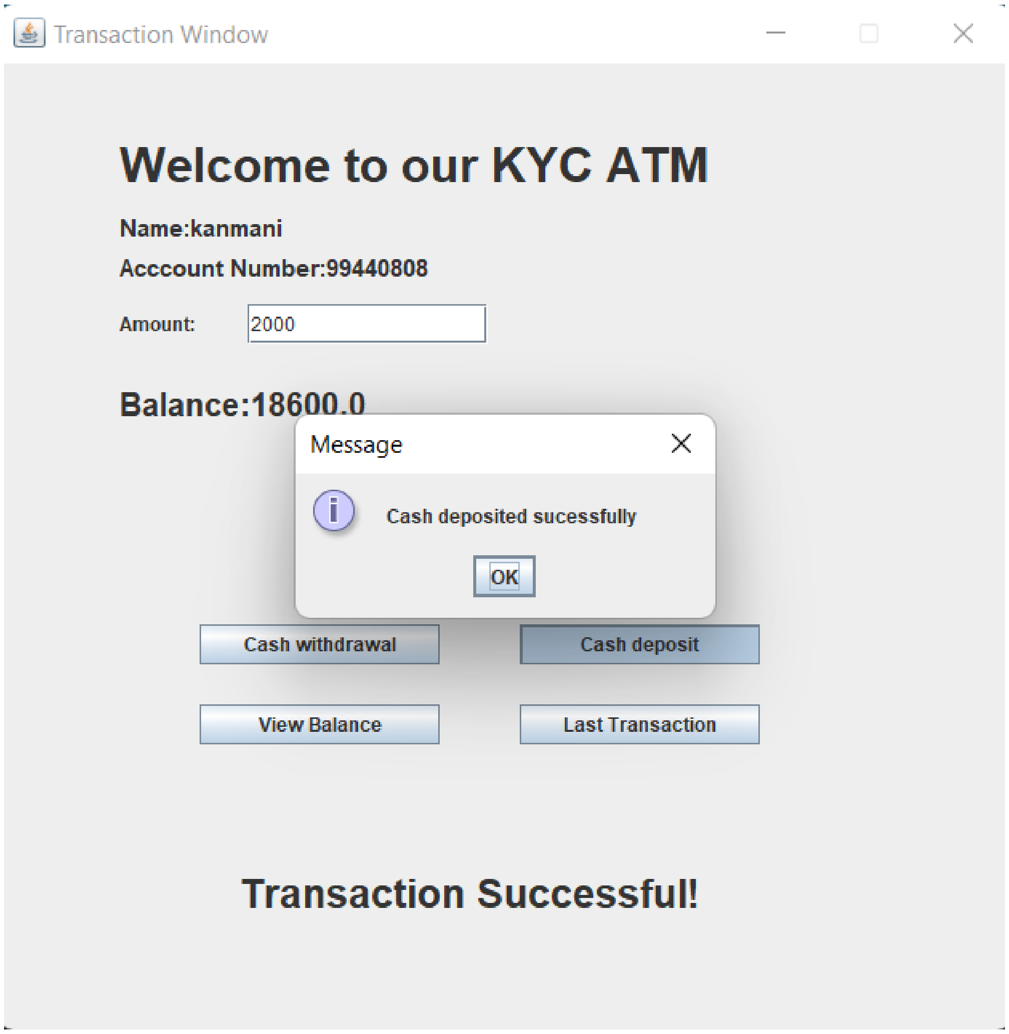
Balance:16600.0

Message

Cash withdrawed sucessfully

Transaction Successful!

If we want to deposit the cash, enter the amount to be deposited and click the cash deposit button. The popup message will appear after the successful transaction:


Transaction Window

Welcome to our KYC ATM

Name:kanmani

Acccount Number:99440808

Amount: 2000

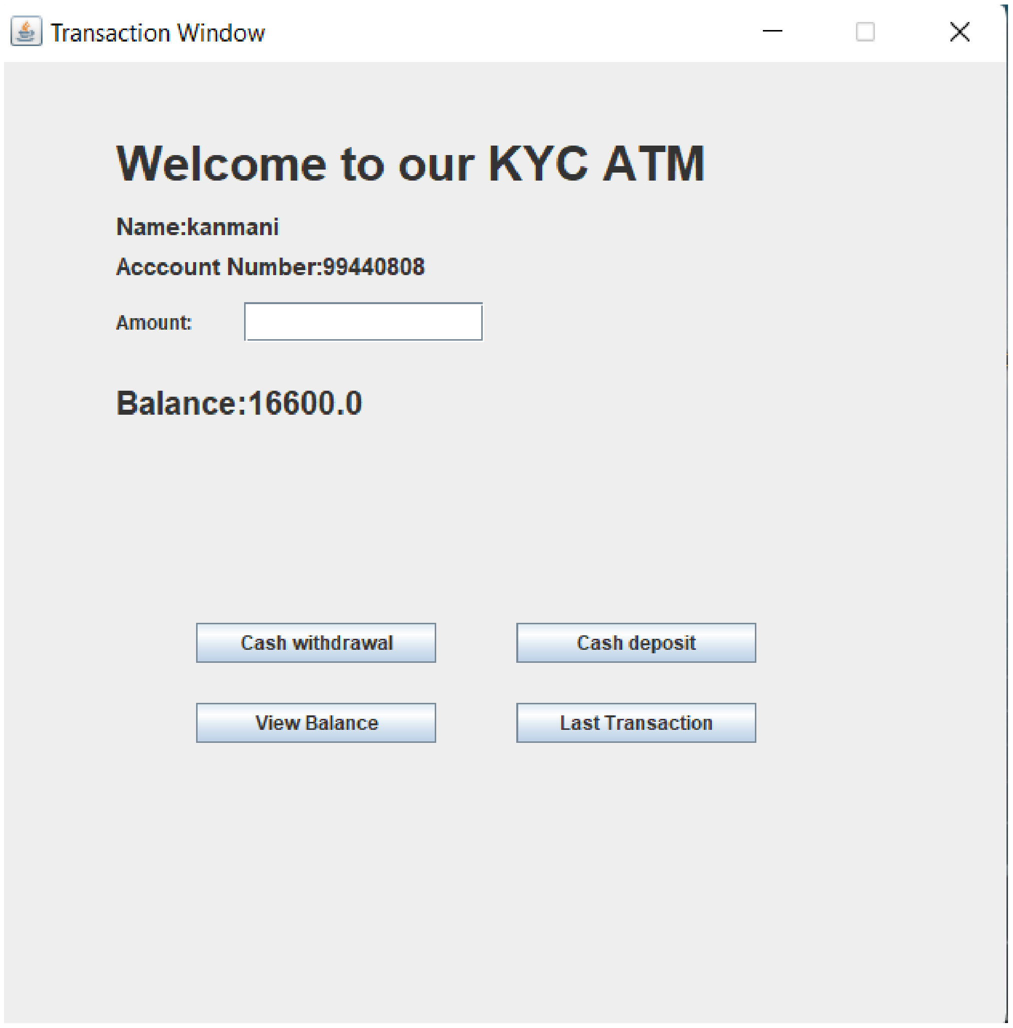
Balance:18600.0

Message

Cash deposited sucessfully

Transaction Successful!

If we want to check the balance, click on view balance button and perform the transaction:


Transaction Window

Welcome to our KYC ATM

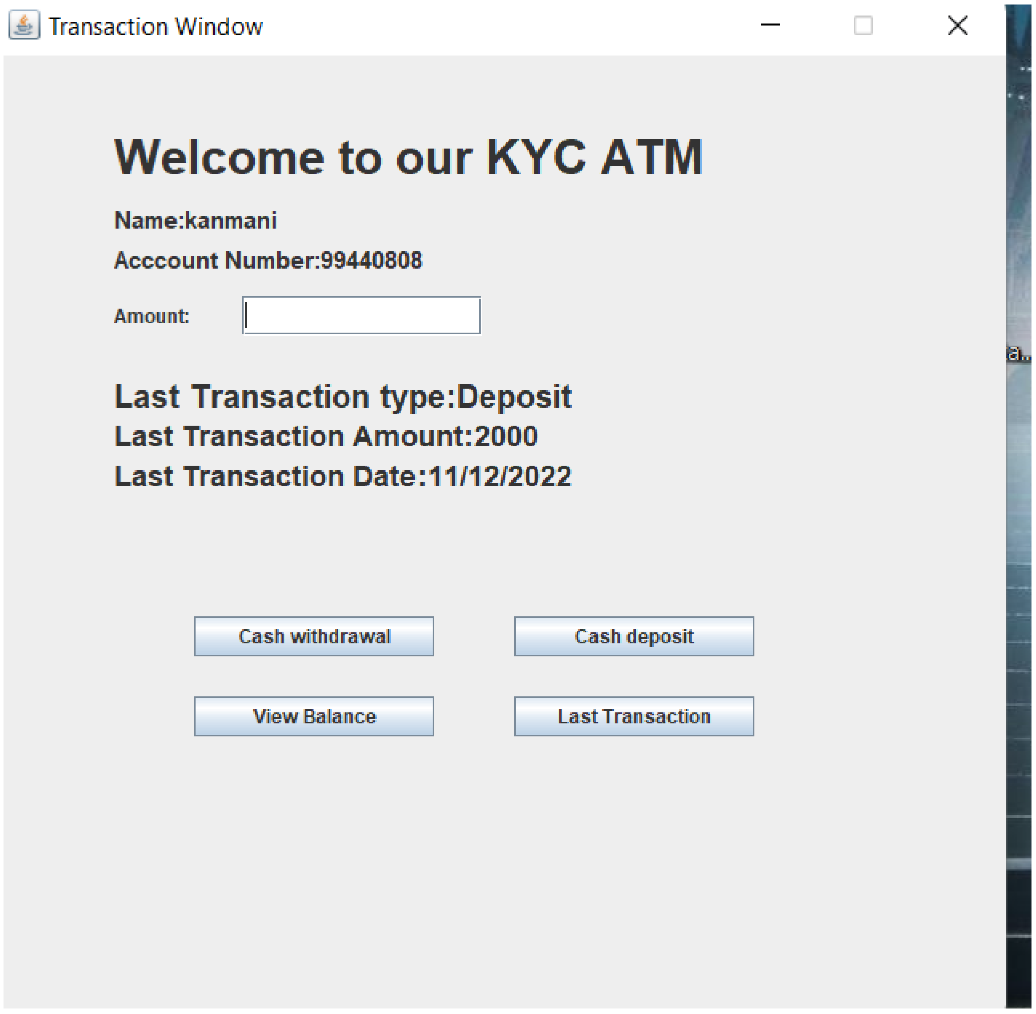
Name:kanmani

Acccount Number:99440808

Amount:

Balance:16600.0

If we want to see the last transaction details, click on last transaction button and perform the transaction:


Transaction Window

Welcome to our KYC ATM

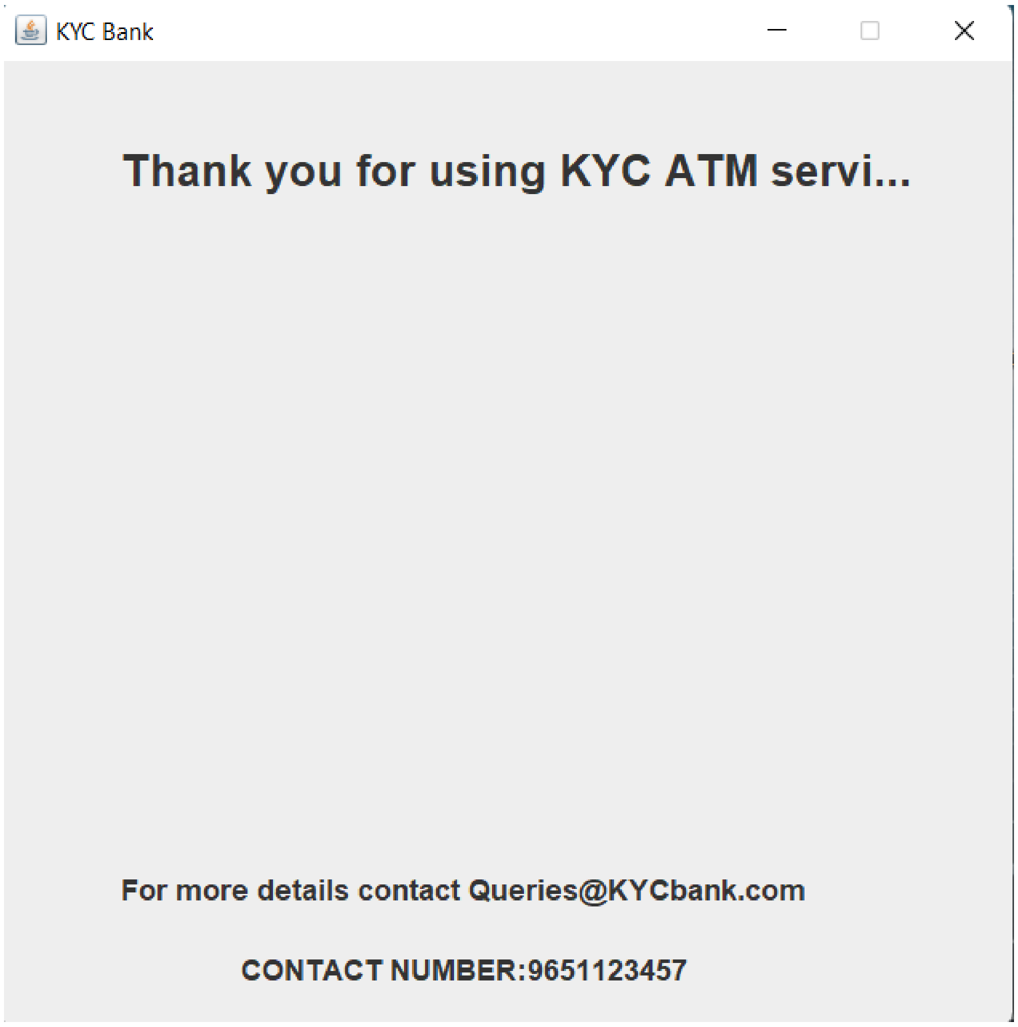
Name:kanmani

Acccount Number:99440808

Amount:

Last Transaction type:Deposit
Last Transaction Amount:2000
Last Transaction Date:11/12/2022

After completing the successful transaction, the end page will be:

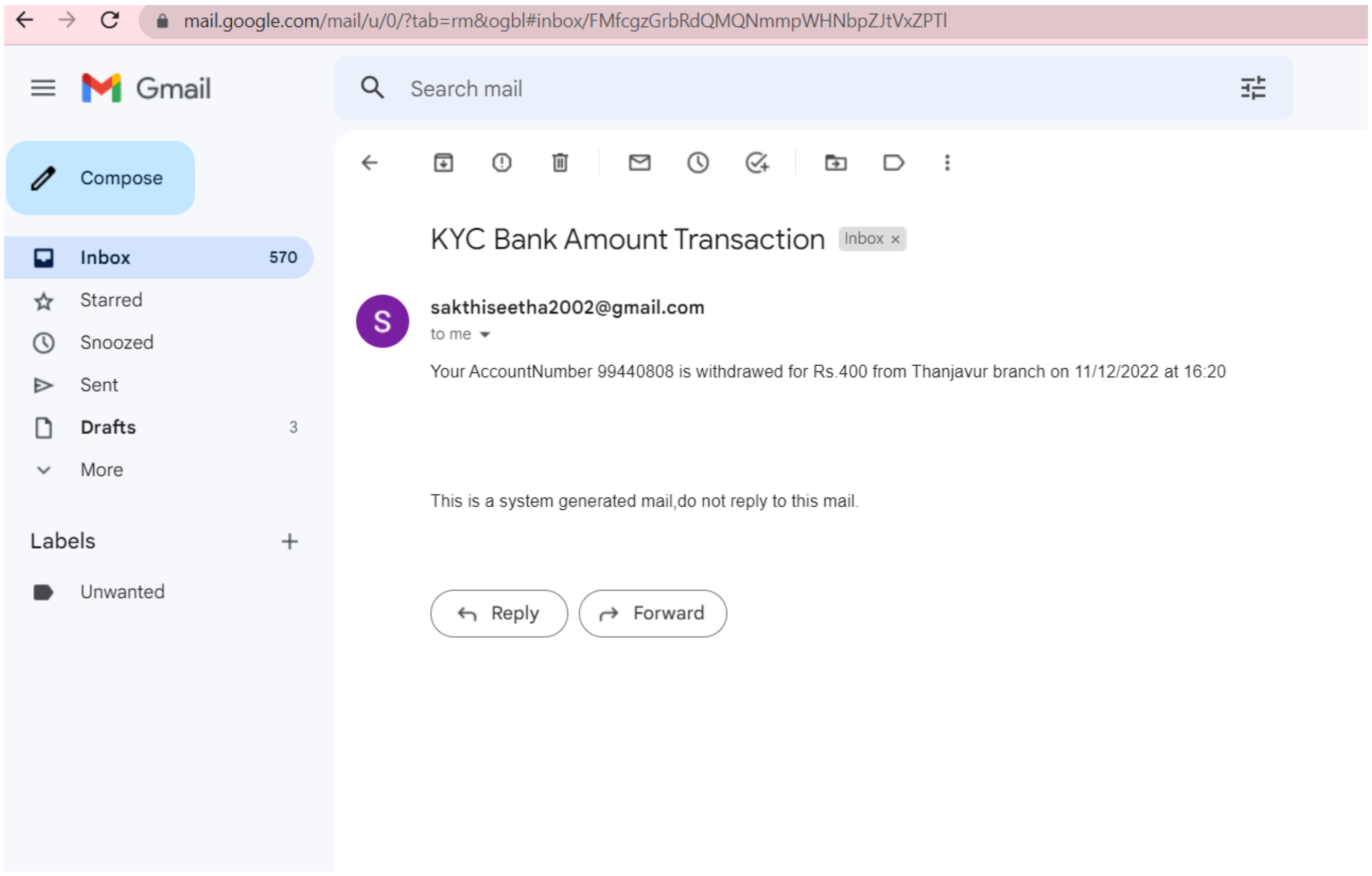


Thank you for using KYC ATM servi...

For more details contact Queries@KYCbank.com

CONTACT NUMBER:9651123457

Whenever the user withdraws or deposits the cash in the ATM machine, he/she will receive an email to their registered email from the bank telling details about their transactions. This will provide additional security since user will come to know whenever the cash is withdrawed or deposited from their account. This is the mail sent from the bank:



KYC Bank Amount Transaction

sakthiseetha2002@gmail.com
to me

Your AccountNumber 99440808 is withdrawed for Rs.400 from Thanjavur branch on 11/12/2022 at 16:20

This is a system generated mail,do not reply to this mail.

CHAPTER 9

CONCLUSION

Thus, ATM machines provides a secure platform to perform financial transactions and helps people to withdraw and deposit money without help of bank representative. the use of physical ATM card can be replaced by digital ATM card that can be stored in our smartphone and when we need to do transaction, we can scan that digital ATM card to machine using Near-Field Communications (NFC) to access our account. This will prevent the physical damage or loss of our ATM card.

CHAPTER 10

REFERENCES

1. Computer Networks and Internets with Internet Applications- 4th Edition by Douglas E. Corner.
2. Java Programming- A Practical Approach by C Xavier – for Java Swing and socket programming.
3. https://www.javatpoint.com/socket-programming. – for Socket Programming https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm. -for AES Encryption and Decryption.
4. https://www.javatpoint.com/steps-to-connect-to-the-database-in-java - for JDBC connection
5. https://www.javatpoint.com/example-of-sending-email-using-java-mail-api - for SMTP