MOBILE RECHARGING WITH BANKING TRANSACTION USING SMS

A PROJECT REPORT

Submitted by

N.NIRMAL RAJ (41501104055)
P.C.PRABHAKARAN (41501104061)
T.RAJA SEKARAN (41501104067)

in partial fulfillment for the award of the degree
of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

S.R.M. ENGINEERING COLLEGE,
KATTANKULATHUR-603 203, KANCHEEPURAM DISTRICT.

ANNA UNIVERSITY : CHENNAI - 600 025

MAY 2005

BONAFIDE CERTIFICATE

Certified that this project report **" MOBILE RECHARGING WITH BANKING TRANSACTION USING SMS "** is the bonafide work of **"N.NIRMAL RAJ(41501104055) P.C.PRABHAKARAN(41501104061) T.RAJA SEKARAN (41501104067) "** who carried out the project work under my supervision.

Prof. SRIDHAR
HEAD OF THE DEPARTMENT
COMPUTER SCIENCE
AND ENGINEERING
S.R.M.Engineering College
Kattankulathur - 603 203
Kancheepuram District.

Mrs.B.SOWMIYA
Lecturer
COMPUTER SCIENCE
AND ENGINEERING
S.R.M.Engineering College
Kattankulathur - 603 203
Kancheepuram District.

ACKNOWLEDGEMENT

A Project is a test of not only technical skills but also teamwork and performance under various constraints. This journey cannot be successfully accomplished without help from experts.

At the outset we would like to thank our director **Dr.T.P.Ganesan**, and principal **Dr.A.Venkataramani** for their support and encouragement. We are also grateful to our Head of the Department **Prof.S.S.Sridhar** who was a great source of inspiration to us during the project work and through out the course.

We will be ever grateful to our guide **Mrs. B.SOWMIYA** without whose guidance, this project would not have become successful.

Finally, we would like to thank our parents and mentors for the support that they have been giving us.

ABSTRACT

The main aim of this project is to provide a customer Friendly service of getting the card recharged and enable the Banking transaction activity by activating transfer of account through SMS. It serves as a helping hand to the people by rendering the service availed to the customer at its level best by skipping traditional customs of on person availability in Banks & other sectors

1. **Mobile Recharge**
2. **Transaction on mobile**

Mobile Recharge:

The SMS must include the details regarding the account number , PIN number and amount for recharging , after this SMS is sent and it passes through mobile receiver where it is checked for validation. After the bank server approves the transaction of deduction in the amount in the bank amount. The request is directed to the mobile

server.The mobile server acts after receiving the intimation from the bank server and successfully recharges the card there by sending the reply from bank to mobile user.

Transaction on mobile:

Transaction on mobile is used to money between two peoples. These two peoples must be registered in a bank. And they should have mobile transaction. This transaction starts with SMS. If user 1 wants to pay Rs.3000 to User 2 .user 1 simply type SMS to particular bank with his 4 Digit PIN, Amount and Account No. the request is processed by the bank server and Amount is transferred to designation account.

CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	…i
	TABLE OF CONTENTS	...ii
1.	**INTRODUCTION**	
	1.1 Information about System	…1
2.	**PROBLEM DEFINITION & METHODOLOGY**	
	2.1 Problem Definition	…2
	2.2 Methodology	...3
3.	**DESIGN DETAILS**	
	3.1 Algorithms	
	3.1.1 Mobile Recharging	...4
	3.1.2 Bank Transaction	…5
	3.2 Dataflow Diagram	...6
	3.3 Architecture	…8

4. LITERATURE REVIEW

4.1 Requirements

4.1.1 Hard wares ...9

4.1.2 Soft wares … 9

4.2 Introduction of Software

4.2.1 Java …10

4.2.2 J2ME Wireless Tool kit 1.0.4 …12

4.3 Wireless Application Protocol ...13

5. IMPLEMENTATION OF MODULES

5.1 Project Description …14

5.2 Explanation of Modules

5.2.1 Bank Server ...15

5.2.2 Mobile Server …15

5.2.3 Mobile Receiver …16

5.2.4 Client Component …16

6. PERFORMANCE EVALUATION …18

7. SYSTEM TESTING …19

8. FUTURE ENHANCEMENTS & APPLICATIONS …22

9. CONCLUSION …23

10. APPENDICES

10.1 Source Code …24
10.2 Sample Input & Output …40
10.3 Bibliography …55

1.INTRODUCTION

The existing system if we want to recharge our mobile mean we have to buy the prepaid recharge card and recharge it. But by our project, just by sending SMS, we can easily recharge the mobile at any where at any time. Similarly, we can transfer the money from one account to another account. This makes the transaction easy.

In order to improve the available facility, we are developing software for both mobile recharging and bank transaction using mobile SMS as wireless communication.

SMS appeared on the wireless scene in 1991in Europe, where digital wireless technology first took root. The European standard for digital wireless now known as the Global Standard for Mobile (GSM) included short message services from the outset.

A distinguish characteristics of the service is that an active mobile handset is able to receive or submit a short message at any time independent of weather or not a voice data call process. SMS is characterized by out-of-band packet delivery and low bandwidth message transfer.

The benefits of SMS to subscriber center around convenience, flexibility and seamless integration of message services and data access. Short message service center is responsible for the relaying and store and forwarding of short message between mobile and mobile station.

2.PROBLEM DEFINITION AND METHODOLOGY

2.1 PROBLEM DEFINITION:

The task under consideration is to provide the mobile recharge and bank transaction facility by accessing the URL.Topic Module Transaction through Internet with the aid of WAP enabled cell phones. The factors to be considered are the low

bandwidth and less transfer rate in the wireless network. So optimized data representation is essential for more efficiency.

The existing system if we want to recharge our mobile mean we have to get the prepaid recharge card and recharge it. But by our project, just by sending SMS, we can easily recharge the mobile at any where at any time. Similarly, we can transfer the money from one account to another account. This makes the transaction easy.

2.2 METHODOLOGY:

This project work on the basis of Wireless Application Protocol (WAP).
WAP is the next generation of new communication transaction. It is an open specification That offers a standard method to access Internet based content and services from wireless ices such as mobile phones.

Wireless mobile kit to send and receive the messages. Mobile receiver acts as a interface between mobile kit and other servers. Database maintenance and administration. Rendering service to the customer an tool kit empowered with Java 2 Micro Edition to represent the model of the process of inflow and outflow of messages. Link must be maintained between mobile kit and servers to hold sway the message and transferring it swiftly to the respective servers by means of filtering. This is done bye mobile receiver.

An process of keeping the personal details and process of transaction of account holders are maintain in a database .It also administers the process of authentication and validation of banking transaction there by updating the relevant information and by executing the task.

Mobile server is the key for offering and effecting the service to the customer. It is natural for it has a database containing the detail of recharge **(**last date and balance amount of account**).** A tabled of data for individual customer is also maintained to envisage the query service.

3. DESIGN DETAILS

3.1 ALGORITHM:

RECHARGING THE MOBILE:

Step 1: Activation the mobile server, bank server and mobile receiver is done.

Step 2: Once these are activated, get the account number, PIN number and the amount to be recharged.

Step 3: The recharge request is sent to the mobile receiver. The request contains information such as account number, PIN number, and amount.

Step 4: The mobile receiver sends the request to bank server for verifying the account number, PIN number and the amount to be recharged.

Step 5: If the information is true then the values in the bank server and mobile sever is updated accordingly.

Step 6: Otherwise, it sends an error message to mobile kit.

Step 7: Terminate the process.

BANK TRANSACTION:

Step 1: Activation the mobile server, bank server and mobile receiver is done.

Step 2: Once these are activated, get the source account number, destination account number, PIN number and the amount to be transferred.

Step 3: The transaction request is sent to the mobile receiver. The request contains information such as source account number, destination account number, PIN number, amount.

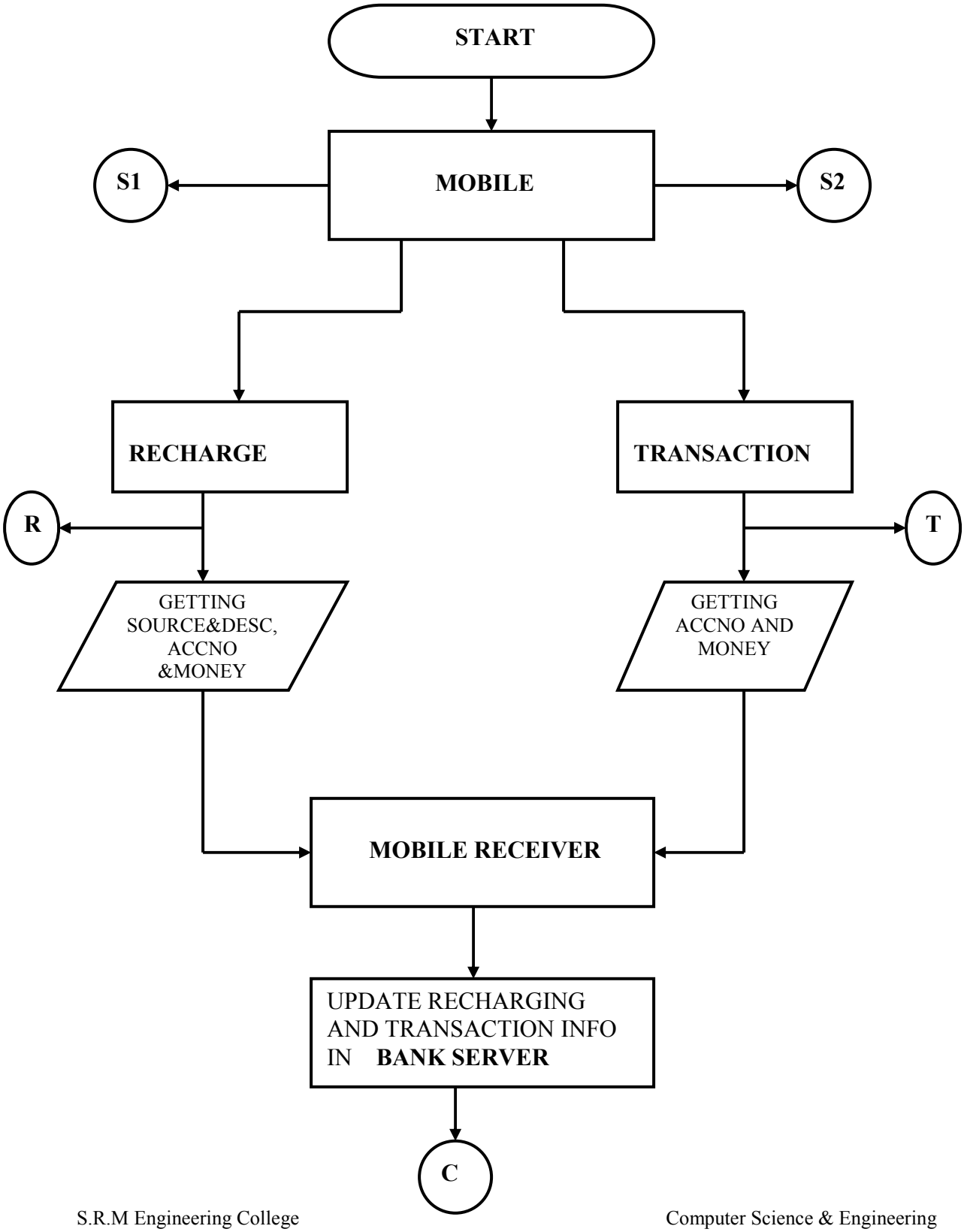
Step 4: The mobile receiver sends the request to bank server for verifying the account number, PIN number and the amount to be transferred.

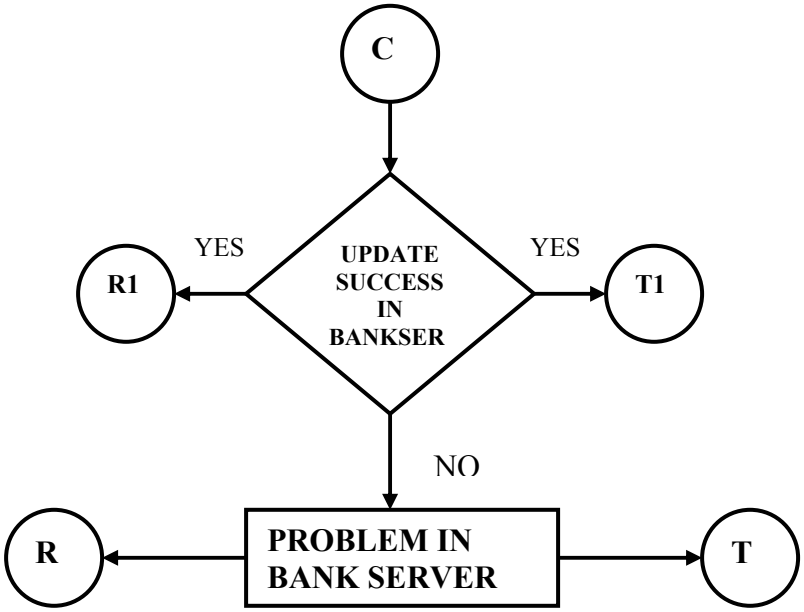
Step 5: If the required amount to be transferred is available in the source account, the bank sever is updated with that in the destination account.

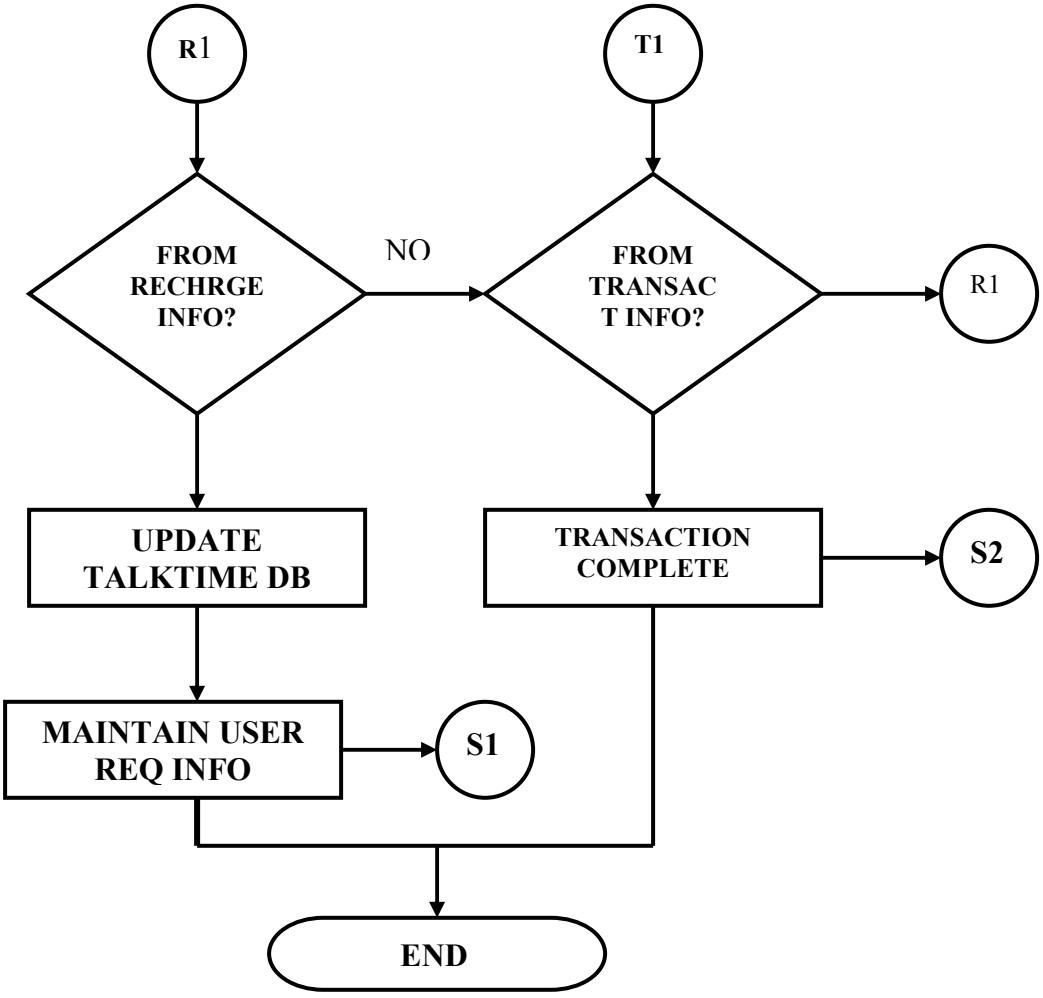
Step 6: If the required amount is not available then it sends the error message to the mobile receiver from which it is displayed in the mobile kit.

Step 7: Terminate the process.

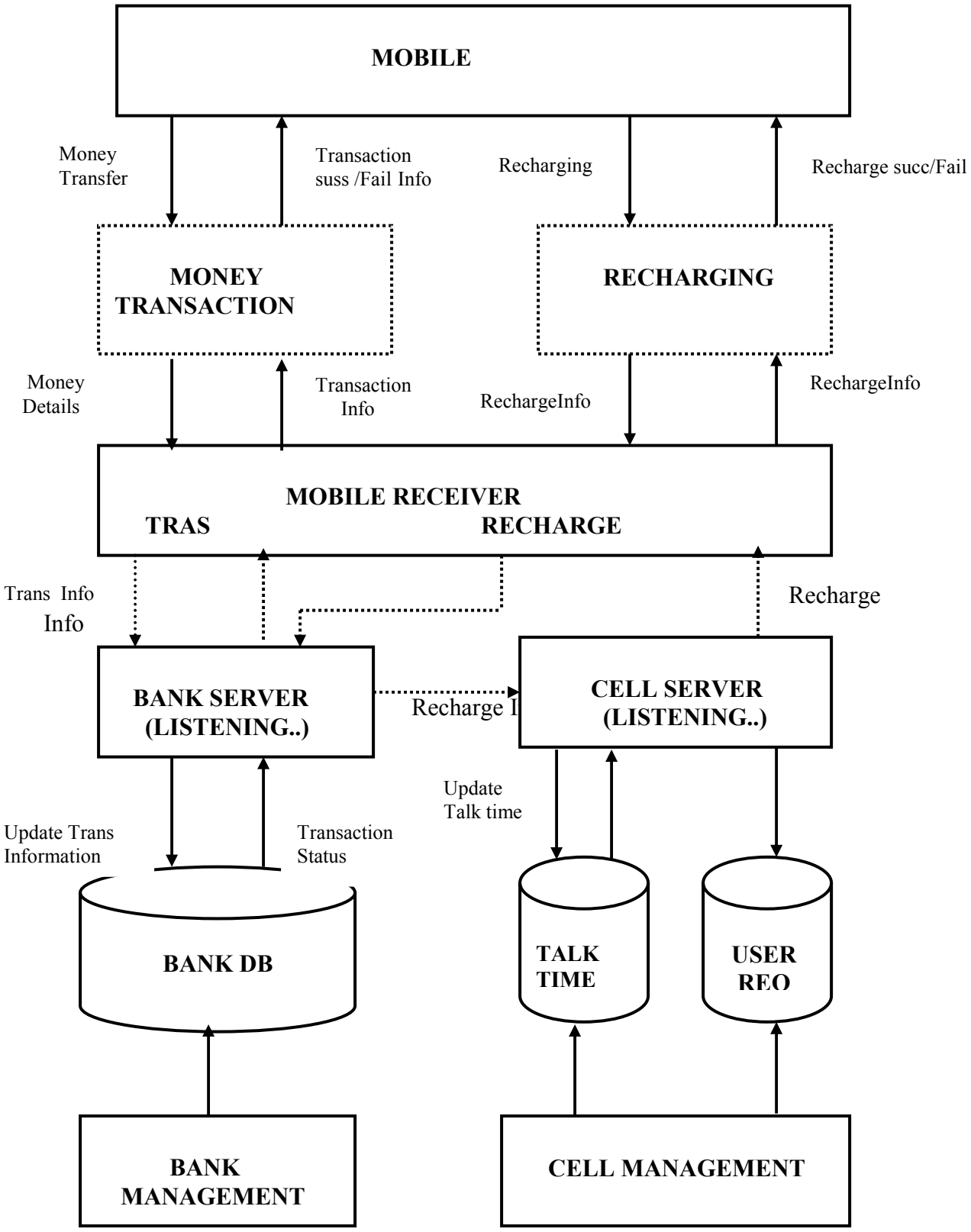
3.2 DATA FLOW DIAGRAM

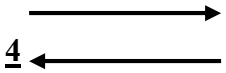



C
YES
UPDATE SUCCESS IN BANKSER
YES
R1
T1
NO
R
PROBLEM IN BANK SERVER
T


R1
T1
FROM RECHRGE INFO?
NO
FROM TRANSAC T INFO?
R1
UPDATE TALKTIME DB
TRANSACTION COMPLETE
S2
MAINTAIN USER REQ INFO
S1
END

3.3 ARCHITECTURE





4

4.1 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

- Processor : Pentium IV 2.5 Ghz
- RAM : 256 MB
- HDD : 40 GB
- FDD : 1.44 MB
- Keyboard : 105 Keys
- Monitor : 14” Soft White color SVGA
- Network Connection

Software Requirements:

- Platform : Windows 98/2000/NT
- Front End : Java JDK1.3, Java Applets, Tom cat 1.4
- Tool : J2ME Wireless Tool Kit 1.0.4
- Back End : Microsoft Access

INTRODUCTION OF SOFTWARE:

FEATURES:

The following features of java language had drawn our attention towards it.

- Java is object oriented; yet it is simple.
- Java is interpreted. You just compile it and run.
- Java applications are portable across multiple platforms.
- Java applications are robust because the runtime system manages memory for us, which is known as automatic garbage collection
- Java's multithreading capability provides the means to build application with many concurrent threads of activities

PLATFROM INDEPENDENT:

Java is independent as it runs on any Operating system. This does not have any direct contact with register.

ROBUST:

Java is highly robust language, which is in the sense, is a typed language. The occurrence of errors within a program is less often.

OBJECT ORIENTED:

Java as its predecessors the Java is an object oriented programming language. Everything in java will be represented as Objects. This helps in data encapsulation.

SECURE:

A language like Java used over an Internet should be so secure that there must be no chance of any security break. If any security lapse occurs then the consequences of it might be disasters to think.

PORTABLE:

As the java suggests, Java programs are portable in the sense that a program written on a machine can be used on another machine irrespective of the architecture of that machine.

SIMPLE:

Java is simple in the sense that if the programmers have little knowledge in C++ then learning Java is very simple for them. The syntax and all the instructions for the loops in java is same as that of C++.

MULTITHREADED:

Multithreading means multitasking within the program. Since Java was designed to meet the world requirement of creating interactively and networked programs, multithreading was included as a part of Java. So java allows the user to write programs that do many things at once.

DYNAMIC:

Java programs carry with them a large amount of information with them that is used to verify and resolve access to the objects at the run time. This makes it possible to dynamically link code in a safe and expedient manner.

DISTRIBUTED:

Java is designed for the distributed environment of the Internet, because it handles TCP/IP protocols. In fact, accessing a resource using a URL is not much different from accessing a file. The original version of Java (Oak) included features for intra address – space messaging. This allowed objects on two different computers to execute procedures remotely. Java revived these interfaces in a package called Remote Method Invocation (RMI). This feature brings an unparallel Client/Server programming.

J2ME WIRELESS TOOL KIT:

The Java TM 2 Platform, Micro Edition, Wireless Toolkit User's Guide describes how to install, configure and work with the J2ME TM Wireless Toolkit and its components.

The J2ME Wireless Toolkit supports the development of Java applications that run on devices compliant with the Mobile Information Device Profile (MIDP), such as Cellular phones, two-way pagers, and palmtops. This document assumes that you are

familiar with Java programming, MIDP, and the Connected Limited Device Configuration (CLDC).

The J2ME Wireless Toolkit supports a number of ways to develop MIDP Applications. You can carry out the development process by running the tools from the command line or by using development environments that automate a large part of this process.

The KToolBar, included with the J2ME Wireless Toolkit, is a minimal development environment with a GUI for compiling, packaging, and executing MIDP applications. The only other tools you need are a third-party editor for your Java source files and a debugger.

An IDE compatible with the J2ME Wireless Toolkit provides even more convenience. For example, when you use the Sun ONE Studio 4, Mobile Edition, (formerly Forte TM for Java TM) you can edit, compile, package, and execute or debug MIDP applications, all within the same environment. For a list of other IDEs that are compatible with the Wireless ToolKit.

INTRODUCTION TO WAP:

WAP (Wireless Application Protocol) is a major breakthrough that achieves universal Internet-based information access on wireless device. It will make it possible for developers to write once for all networks worldwide. Carriers will be able to implement gateways that work with many brands of phones and all applications and content. Handset manufactures can make high volume, low cost handsets for all carriers.

The purpose of Wireless Application Protocol is to provide operators, infrastructure and terminal manufacturers, and content developers a common environment that will enable development of value-added services for mobile phones. The four founding members aim to create, together with other industry partners, a global wireless service specification to be adopted by appropriate standards bodies and will be independent of the network infrastructure system in place. All the applications of the protocol will be scaleable regardless of transport option and device types.

The Wireless Application Protocol is target to bring advanced service and Internet content to digital cellular phones and terminals. A common standard means the potential for realizing economics of scale, encouraging cellular network carriers to develop new differentiated service offering as a way attracting new subscribers. Consumers benefit through more and varied choice in advance mobile communications applications and service.

4.2 IMPLEMENTATION OF MODULES

PROJECT DESCRIPTION:

The project entitled " Mobile recharging system with Banking Transaction" is developed using JAVA &Tom cot Server as its front end. The main aim of this project is to provide an customer Friendly service of getting the card recharged and enable the Banking transaction activity by activating transfer of account through SMS. It serves as an helping hand to the people by rendering the service availed to the customer at its level best by skipping traditional customs of on person availability in Banks & other sectors.

MODULES:

- Bank server
- Mobile server
- Mobile receiver
- Client component.

EXPLANATION OF MODULES:

Bank server:

It maintains an database of individual account information by an Bank customer. This in calculates accounts, security code i.e., Personal Identification Number along with its name, memorandum and personal details of individual, type of account. The database is flexible and amand itself instanteously on a transaction being executed. Once after recharging, the databases get updated and keep a regular watch on the accounts and informs the celcom server to endorse the request.

Mobile server:

It also maintains two database table regarding user information and talk time. Cellular operators manage the data base table. It receives the request after the processing of all these valid data's from the mobile kit to the mobile server. Here the request is given green signal and the database is refreshed with the new data, which adds upon every time card is recharged.

They change in accordance to users talk time value. This is maintained in the data table. It also includes details regarding mobile number, amount still existing in the card and date of last renewal.

The other table administrated by the mobile server is most prominent called mobile data base. It furnishes details of the primary mobile number, secondary mobile number, amount and date of renewals of the particular mobile number.

The contrasts between mobile and data table is the mobile table gives an gross individual values of a particulars user. The date for each refreshable is also notified in the mobile table.

Mobile receiver:

It an interface where we above to receive the messages from the mobile kit. It has an direct link with bank server. Once the message from the kit is received, this server gets initialized and automatically passes the request to the bank server for verifying the facts

involving account number, pin number, amount. Once the details are checked, if it true the quoted recharge amount is updated with the existing ones.

Client component (Mobile kit):

It is usually called as client component is an toolkit used to send and receive message by means of simple message transfer. It incorporates J2ME wireless toolkit features. In order to do the process of recharging we are suppose the send the message in the of message service to the toll number. The cellular operator renders the service of recharging and transaction in the schemes in accordance to the connectivity respectively.

Mobile recharging:

The SMS must include the details regarding the account number , PIN number and amount for recharging , after this SMS is sent and it passes through mobile receiver where it is checked for validation. After the bank server approves the transaction of deduction in the amount in the bank amount. The request is directed to the mobile server.The mobile server acts after receiving the intimation from the bank server and successfully recharges the card there by sending the reply from bank to mobile user.

Bank transaction:

The SMS must include the details regarding the source account number, destination account number , PIN number and amount to be transferred, after this SMS is sent and it passes through mobile receiver where it is checked for validation. After the bank server approves the transaction of deduction in the amount in the bank amount. The destination amount is updated with the amount to be transferred. It sends reply to the mobile that transaction completed successfully.

6. PERFORMANCE EVALUATION

The various transaction processes in cell phone were performed and the corresponding status of information was reflected at the back end. The transaction information was performed automatically from service provider to user. The master database is placed in the server. It can give access to any number of authorized users. It improves the reliability of transaction. Since information is automatically send to the user.

7.TESTING

System testing is the stage of implementation, it is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding possible errors. A good test case is the one that has high probability of finding a yet undiscovered error. A successful test is one that uncovers a yet undiscovered error

Testing a vital to the success of the system .System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to variety if tests on-line response; volume, stress, recovery, and the system security and usability tests. A series of tests are performed before the system is ready for user acceptance testing.

TYPES OF TESTING:

The different types of testing are

1. Unit testing
2. Integrated testing
3. Black box testing
4. Validation
5. Output testing

UNIT TESTING:

In this testing, we test each module individual and integrated the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as 'module testing' .The modules of the system are tested separately. This testing is carried out during programming stage itself. In this step, each module is found working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the date input given by the user, which both the format and validity of the data entered. It is very easy to find error debug the system.

INTEGRATED TESTING:

Data can be lost across an interface, one module can have an adverse effect on the other sub functions when combined, may not produce the desired major functions. Integrated testing is systematic testing for constructing the uncovers errors within the interface. This testing is done with simple data. The need for integrating test is to find the over all system performance.

BLACK BOX TESTING:

This testing is done to find

1. Incorrect or missing functions
2. Interface errors
3. Errors in external database access
4. Performance errors
5. Initialization and termination errors

The above testing was successfully carried out for these applications according to the user requirement specification.

TEST DATA AND OUTPUT:

Taking various kinds of data does the above testing preparation of test data plays vital role in system testing. After preplanning the test data, system under study is testing using test data. While testing the system with data. Errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

The system has been verified and validated by running

1. Test data
2. Live data

At first the system was tested with some simple test data that are generated by me with my knowledge of the possible range that are required to hold the by the fields. The system ran successfully with the given test data. Then I gathered one month's live data from pass section. While doing this live data testing few problems across which are rectified by me later.

8. FUTURE ENCHANCEMENTS AND APPLICATIONS

FUTURE ENCHANCEMENTS

- To foresee the balance amount in the bank.
- It can be implemented to have credit & debit based transaction.
- Footing the bill for electricity, telephone, taxes that will be deduced at banks by means of SMS.
- Providing valuable information to the customer about the new arrival of schemes in the company and offers rendered by the company.
- Reserving tickets in modes of transport & booking made in the theatres via SMS.
- Reporting the stock values, hot news and sending remainders.

APPLICATIONS:

- It can be applied for using the major cellular operations to provide user-friendly services.
- It offers valuable services in the area where the operators are out of reach.
- There is no geographical bar on its usage.
- It can be accessed at "any where at any time" money transfer & mobile recharge.

9. CONCLUSION

Through the Internet transaction facilities are improved the mobile transaction. It is restricted statically any specific area. The project has completed successfully and the result is verified with various inputs under various circumstances. The goal of our project is reached i.e., recharging the mobile and bank transaction through SMS. Thus the facilities on the mobile phone would makes transactions easy, time saving.

10. APPENDIX

10.1 SOURCE CODE

/*BANK SERVER*/

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.util.*;
import java.sql.*;
public class BankServer extends Frame implements Runnable  {
    public void init()    {
      setTitle("Bank Server Application:");
      setVisible(true);
      setLayout(null);
```

```
        label1.setText("Bank server Running....");
        add(label1);
        label1.setFont(new Font("Dialog", Font.PLAIN, 25));
        label1.setBounds(156,72,300,40);
       txtArea.setBounds(36,132,504,252);
      add(txtArea);
   try       {
     server = new ServerSocket(5000);
     txtArea.append("\n"+"BankServer :Server Loaded....");
     thread = new Thread(this);
     thread.start();     }


  public void run() {
    try    {
        System.out.println("BankServer : waiting for Client Request");
         txtArea.append("\n"+"BankServer : waiting for Client Request");
        socket = server.accept();
       System.out.println("BankServer : Client Connected");
       txtArea.append("\n"+"BankServer : Client Connected");
      BankHandler bankHandler = new
  BankHandler(socket.getInputStream(),socket.getOutputStream());
  bankHandler.start();
  }catch(Exception ex)   {
        txtArea.append("\n"+"BankServer : " + ex);              }         }
  public static void main(String arg[])               {
        BankServer b=new BankServer();
        b.init();  }
public TextArea txtArea = new TextArea();
ServerSocket server = null;
Socket socket = null;
```

```
    Thread thread = null;
    Label label1 = new Label();
          class BankHandler extends Thread   {
           private DataOutputStream dos = null;
    private String strACNO=null,strAMT=null,strPIN=null,strState1=null,strState2=null;
          public  int amount,cardamount;
          public BankHandler(){}
          public BankHandler(InputStream in,OutputStream out) {
          try{
          dis = new DataInputStream(in);

          dos = new DataOutputStream(out);}
          catch(Exception ex){
                    txtArea.append("\n"+"BankServer : "+ex);}
        }
public void run(){
try{
while(true){
          txtArea.append("\n"+"BankServer :waiting for read message" );
          String strReaded = dis.readUTF();
          if(strReaded != null){
    System.out.println("BankServer : Message Received <<" + strReaded.trim() +">>");
     txtArea.append("BankServer : Message Received <<" + strReaded.trim()+">>\n");
     if(!strReaded.trim().equals("Recharged Successfully"))     {
       StringTokenizer token = new StringTokenizer(strReaded.trim(),",");
      strAMT = token.nextToken();
      strState1=token.nextToken();
      Connection c = null;
      PreparedStatement  s,s1,s2,s3 ;
      ResultSet r,r1;
      Socket soc;
```

```
      DataInputStream din;
     DataOutputStream dout;
try  {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   c=DriverManager.getConnection("jdbc:odbc:reg","","");}
catch(Exception o)  {
     txtArea.append("\n"+o.getMessage()); }
if(strState1.equals("1111")) {
 System.out.println("statte1:"+strState1);
     txtArea.append("\n"+"Acount No:"+strACNO);
      txtArea.append("\n"+"PINNo:"+strPIN);
      txtArea.append("\n"+"Amount:"+strAMT);
          s=c.prepareStatement("select Amount,Mobile from
             Register where AccNo=? andPinNo=?");
          s.setString(1,strACNO);
          s.setString(2,strPIN);
          s1=c.prepareStatement("update Register SET Amount=?
                where AccNo=? and PinNo=?");
          r=s.executeQuery();
          soc=new Socket("localhost",6000);
          InputStream in=soc.getInputStream();
          OutputStream out=soc.getOutputStream();
          dout=new DataOutputStream(out);
          din=new DataInputStream(in);
          if(r.next())  {
                    String amt=r.getString("Amount");
                    String cell=r.getString("Mobile");
                    amount=Integer.parseInt(amt);
                    cardamount=Integer.parseInt(strAMT);
          try{      if(amount<cardamount){
                              dout.writeUTF("You Don't Have Enough Money");}
```

```
                    else{
                            int amountnew=amount-cardamount;
                            Integer i6=new Integer(amountnew);
                           String amtstrnew=i6.toString();
                            s1.setString(1,amtstrnew);
dout.writeUTF("Account
Success"+","+strAMT+","+strPIN+","+strACNO+","+cell);}
          din=new DataInputStream(soc.getInputStream());   }
 catch(Exception e)              {
              txtArea.append(e.getMessage());
             dout.writeUTF("Check Your ACCNO and PINNO");       }
}
          System.out.println(strACNO);
          System.out.println(strPIN);
          System.out.println(strAMT);
}
          else     {
                    System.out.println("statte1:"+strState1);
                    txtArea.append("\n"+" SRC AccNo:"+strACNO);
                    txtArea.append("\n"+"DEC AccNo:"+strPIN);
                    txtArea.append("\n"+"Amount:"+strAMT);
                    txtArea.append("\n"+"pin No"+strState1);
                    System.out.println("one");
                    s=c.prepareStatement("select Amount,Mobile from
                     Register where AccNo=? and PinNo=?");
                    s.setString(1,strACNO);
                    s.setString(2,strState1);
                    System.out.println("two");
                    s1=c.prepareStatement("update Register SET Amount=?
                          where AccNo=? and PinNo=?");
                    s2=c.prepareStatement("update Register SET Amount=?
```

```
        where AccNo=?");
    s3=c.prepareStatement("select Amount from Register
        where AccNo=?");

    soc=new Socket("localhost",6000);
    InputStream in=soc.getInputStream();
    OutputStream out=soc.getOutputStream();
    dout=new DataOutputStream(out);
    din=new DataInputStream(in);
    if(r.next()) {
    String amt=r.getString("Amount");
    System.out.println("First Amount:"+amt);
    String cell=r.getString("Mobile");
    amount=Integer.parseInt(amt);
    cardamount=Integer.parseInt(strAMT);
    try {
    if(amount<cardamount) {
            dout.writeUTF("You Don't Have Enough Money");
}
    else    {
            int amountnew=amount-cardamount;
            System.out.println("After Amount:"+amountnew);
            Integer i6=new Integer(amountnew);
            String amtstrnew=i6.toString();
            s1.setString(1,amtstrnew);
            s1.setString(2,strACNO);
            dout.writeUTF("AccountSuccess"+","+strAMT+",
            "+strPIN+","+strACNO+","+cell);
            r1=s3.executeQuery();
            if(r1.next())    {
            String decamount=r1.getString("Amount");
```

```
                    int decamount1=Integer.parseInt(decamount);
                    int dectotamount=decamount1+Integer.parseInt(strAMT);

                    Integer dec=new Integer(dectotamount);
                    String dec1=dec.toString();
                    s2.setString(1,dec1);
                    s2.setString(2,strPIN);
                    s2.executeUpdate();
                    System.out.println("Transaction Completed");
                    }       }
                    din=new DataInputStream(soc.getInputStream());
            }
            catch(Exception e)    {
                    txtArea.append(e.getMessage());
                    dout.writeUTF("Check Your ACCNO and PINNO");
                    } }
                    System.out.println(strACNO);
                    System.out.println(strPIN);
                    System.out.println(strAMT);  }
                    dos.writeUTF("B0");  }
                else    {
                            System.out.println("Recharged Sucessfully");
                    }       }
            else    {
                dos.writeUTF("BN");         }       }   }
            catch(Exception ex)   {

                    txtArea.append("\n"+"BankServer :"+ex);    }
    } } }
```

/*MOBILE SERVER*/

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.util.*;
import java.sql.*;
public class CelcomServer extends Frame implements Runnable   {
        public TextArea txtArea = new TextArea();
        Label label1=new Label();
        private ServerSocket server = null;
        private Socket socket = null;
        Thread thread = null;
        public void init()  {
            setTitle("Celcom Server Application:");
            setVisible(true);
            setLayout(null);
            label1.setText("Celcom server Running....");
            add(label1);
           label1.setFont(new Font("Dialog", Font.PLAIN, 25));
            label1.setBounds(156,72,300,40);
            setBackground(new java.awt.Color(199,199,226));
            setSize(550,480);
            txtArea.setBounds(36,132,504,252);
            add(txtArea);
         try  {
                server = new ServerSocket(6000);
                thread = new Thread(this);

                thread.start();
```

```
}catch(Exception ex){
                txtArea.append("\n"+"Celcomserver: "+ex);}
}
public void run(){
        try {
           txtArea.append("\n"+"CelcomServer:waitingforClientRequest");
           socket = server.accept();
           txtArea.append("\n"+"CelcomServer : Client Connected");
           CelcomHandler celcomHandler = new CelcomHandler
           (socket.getInputStream(),socket.getOutputStream());
           celcomHandler.start();
        }catch(Exception ex)    {
              txtArea.append("\n"+"CelcomServer : "+ex);
            }}
        public static void main(String arg[])   {
            CelcomServer c=new CelcomServer();
            c.init();
          }
            class CelcomHandler extends Thread{
                  private String strMsg = null;
                   private String strACNO=null,strAMT=null,
                   strPIN=null,strMobile=null;
                   public CelcomHandler(){}
        public CelcomHandler(InputStream in,OutputStream out){
                try   {
                      dis = new DataInputStream(in);
                      dos = new DataOutputStream(out);
                    }
                catch(Exception ex)      {
                     txtArea.append("\n"+"CelcomServer : "+ex);    }
                }
```

```
public void run()     {
   try       {
         while(true)    {
   System.out.println("CelcomServer:waitingforreadmessage");
   String strReaded = dis.readUTF();
   System.out.println("Cellcom server:"+strReaded);
System.out.println("Message:"+strReaded);
txtArea.append("\n"+"Message :"+strReaded);
StringTokenizer token1 = new StringTokenizer(strReaded.trim(),",");
strMsg=token1.nextToken();
strAMT = token1.nextToken();
txtArea.append("\n"+strAMT);
strPIN = token1.nextToken();
txtArea.append("\n"+strMobile);
   Connection c,c1,c2,c3;
   PreparedStatement  s,s2,s1,s3,s4,s5,s6 ;
   ResultSet r,r1,r2,r3,rs4;
     try    {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");    }
catch(Exception h){}
c=DriverManager.getConnection("jdbc:odbc:aaa","","");
s=c.prepareStatement("insert into Mobile values(?,?,?,?,?,?)");
 s1=c.prepareStatement("selectPriMobileNo,Amount,AccNofromMobilew
                        hereAccNo=?");
  s2=c.prepareStatement("insert into Data values(?,?,?,?)");

s3=c.prepareStatement("select Amount from data where MobileNo=?");
s4=c.prepareStatement("update data SET Amount=? where MobileNo=?");
 s5=c.prepareStatement("delete * from data where MobileNo=?");
 s6=c.prepareStatement("insert into data values(?,?,?,?)");
  java.util.Date d=new java.util.Date();
```

```
s.setString(1,strMobile);
s.setString(2,d.toString());
s.setString(3,strAMT);
s.setString(4,"ICICI");
s.setString(5,strACNO);
s.setString(6,strMobile);
s.executeUpdate();
txtArea.append("\n"+"Sucessful");
s1.setString(1,strACNO);
r1=s1.executeQuery();
if(r1.next())    {
        String mobi=r1.getString("PriMobileNo");
        String amt=r1.getString("Amount");
        String Ano=r1.getString("AccNo");
        java.util.Date d1=new java.util.Date();
        s2.setString(1,mobi);
        s2.setString(2,amt);
        s2.setString(3,Ano);
        s2.setString(4,d.toString());
        s2.executeUpdate();
        txtArea.append("succ1");
        s3.setString(1,mobi);
        r2=s3.executeQuery();
        if(r2.next())  {

          String amt1=r2.getString("Amount");
              int amount=Integer.parseInt(amt1);
              int amount1=Integer.parseInt(strAMT);
              int total=amount+amount1;
              Integer ans=new Integer(total);
        String ans1=ans.toString();
```

```
                    s4.setString(1,ans1);
                    s4.setString(2,mobi);
                    s4.executeUpdate();
                    s5.setString(1,mobi);
                    s5.executeUpdate();
                    txtArea.append("Sucessfully Updated with Existing Amount");
                }}
                    Socket soc=new Socket("localhost",9595);
                    InputStream in=soc.getInputStream();
                    OutputStream out=soc.getOutputStream();
                    if(strReaded != null)   {
                        System.out.println("CelcomServer : Message Received <<"
                                            + strReaded.trim() +">>");
                       txtArea.append("\n"+"CelcomServer : Message Received <<"
                                            + strReaded.trim()+">>\n");
                     String success="Recharged Successfully";
                                            out.write(success.getBytes());}
                     else   {
                            txtArea.append("Recharged Successfully");
                             String failure="You Are not a Valid User";
                           out.write(failure.getBytes());}}
                }
            }}
```

/*MOBILE RECIEVER*/

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.util.*;
```

```
public class TCP01Client extends Frame implements Runnable{
        private ServerSocket server=null;
        public  TextArea txtArea = new TextArea();
        Label label1=new Label();
        public static Socket socket=null;
        private Socket celcomsocket = null;
        private Thread thread = null;
        private TCP01Handler tcp01 = null;
        private TCP01Handler tcp02 = null;
        public void init()                  {
                        setTitle("Mobile Receiver Application:");
                        setVisible(true);
                        setLayout(null);
                        label1.setText("Mobile Receiver Running....");
                        add(label1);
                        label1.setFont(new Font("Dialog", Font.PLAIN, 25));
                        label1.setBounds(156,72,300,40);
                        setBackground(new java.awt.Color(199,199,226));
                        setSize(550,480);
                        txtArea.setBounds(36,132,504,252);
                        thread.start();             }
                    public void run() {
                        try      {
                        txtArea.append("\n"+"TCP01Client : Connecting to Bank
                        Server....");
                        socket = new Socket("localhost",5000);
                        txtArea.append("\n"+"TCP01Client : Connected with 5000 Bank
                        Server....");
                        tcp01 = new
                TCP01Handler(socket.getInputStream(),socket.getOutputStream());
                }catch(Exception ex)            {
```

```
                    txtArea.append("\n"+"TCP01Client : "+ex);          }
        try         {
                    server = new ServerSocket(9595);
                    txtArea.append("\n"+"Mobile Reciever Running at 9595");
                }catch(Exception ex){
                    txtArea.append("\n"+"Errror to Bind 9595....");          }
                try         {
                while(true){
                    txtArea.append("\n"+"TCP01Client : waiting for Mobile
                    Connect....");
                    socket = server.accept();
                txtArea.append("\n"+"TCP01Client : waiting for Mobile Connected....");
                    new SendAndReceive(socket.getInputStream(),
                    socket.getOutputStream(),tcp01).start();
                    }
                }catch(Exception ex) {}      }

public static void main(String arg[])  {
        TCP01Client t=new TCP01Client();
        t.init(); }
class SendAndReceive extends Thread{
        TCP01Handler tcp01 = null;
      OutputStream out = null;
        InputStream  in  = null;
      byte b[] = new byte[160];
        public SendAndReceive(InputStream in,OutputStream out,TCP01Handler tcp01){
          this.tcp01 = tcp01;
          this.in = in;
          this.out = out;         }
public void run(){
          try  {
```

```
            in.read(b);
            txtArea.append("\n"+"SendAndReceive : Message <<" +  new
          String(b).trim()+">>");
          String info="Recharged Successfully";
          String strReceived = tcp01.sendNReceive(new String(b));
          txtArea.append("\n"+"SendAndReceive : Received <<" + strReceived.trim()
+">>");  } catch(Exception ex){
          System.out.println("SendAndReceive : " + ex);
        txtArea.append("\n"+"SendAndReceive : " + ex);       }}}
        class TCP01Handler extends Thread{
          private String strMsg = null;
          public TCP01Handler(InputStream in,OutputStream out)     {
          try       {
          dis = new DataInputStream(in);

          dos = new DataOutputStream(out);
          txtArea.append("\n"+"TCP01Client : Init");  }
          catch(Exception ex)               {
                    txtArea.append("\n"+"TCP01Client : "+ex); }         }
          public String sendNReceive(String strMessage) throws Exception{
                    this.strMessage = strMessage;
                    txtArea.append("\n"+"TCP STARTED >>>>>>>>> ");
                    try                 {
                    dos.writeUTF(strMessage);
                    txtArea.append("\n"+"TCP01Client : Message Send"+strMessage);
                    txtArea.append("\n"+"TCP01Client : Waiting for Read");
                    String strReaded = dis.readUTF();
                    if(strReaded != null)             {
                      txtArea.append("TCP01Client : Message Received :
                    <<"+strReaded.trim()+">>\n");                   }
                    return strReaded;
```

```
                }catch(Exception ex)            {
                txtArea.append("TCP01Client : " +ex);
                throw ex;       }       }
        public void run()       {
        txtArea.append("TCP STARTED >>>>>>>>>> ");
                try             {
                dos.writeUTF("1222222,1111,200");
                txtArea.append("TCP01Client : Message Send");
                txtArea.append("\n"+"TCP01Client : Message
Received"+strReaded+"\n");
                }catch(Exception ex)            {
                System.out.println("TCP01Client : " +ex);
                txtArea.append("TCP01Client :"+ex); } } }}
```

10.2 SAMPLE INPUT & OUTPUT

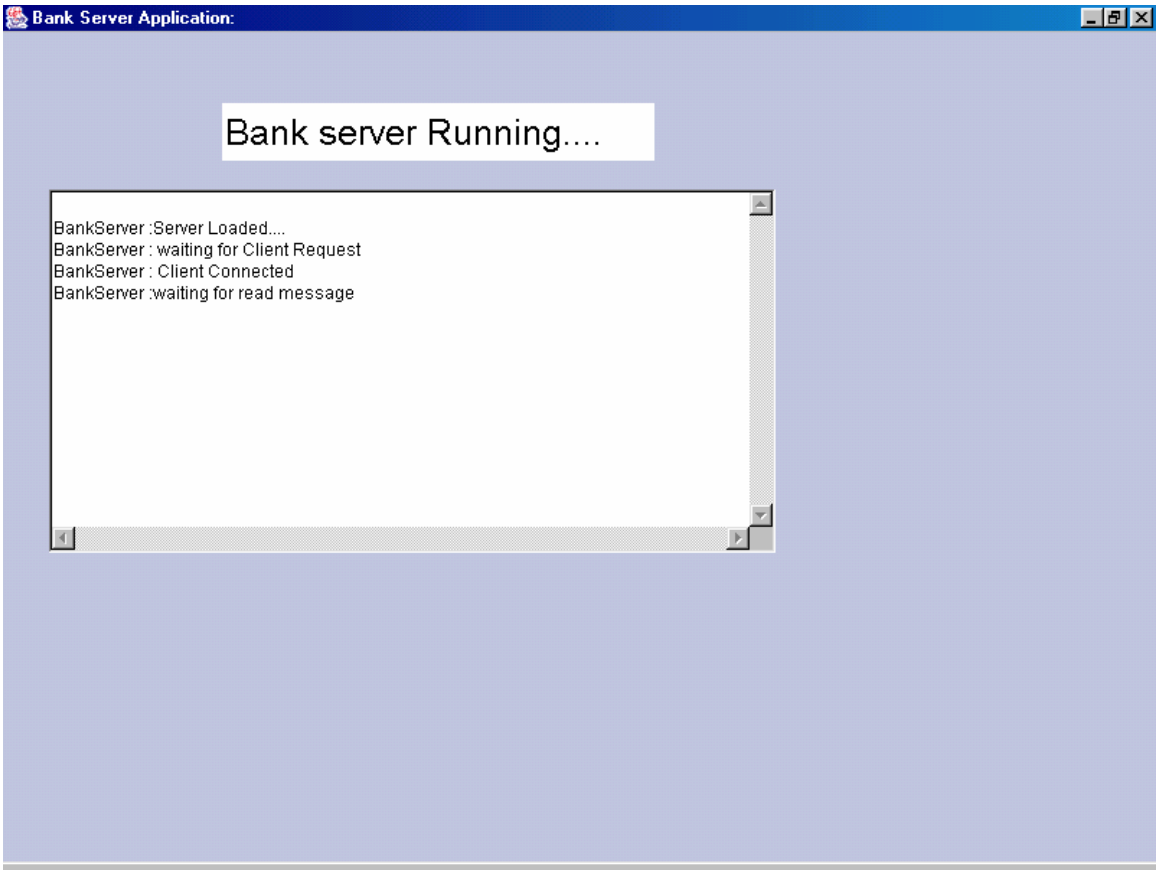


Fig 1. Shows bank server running.

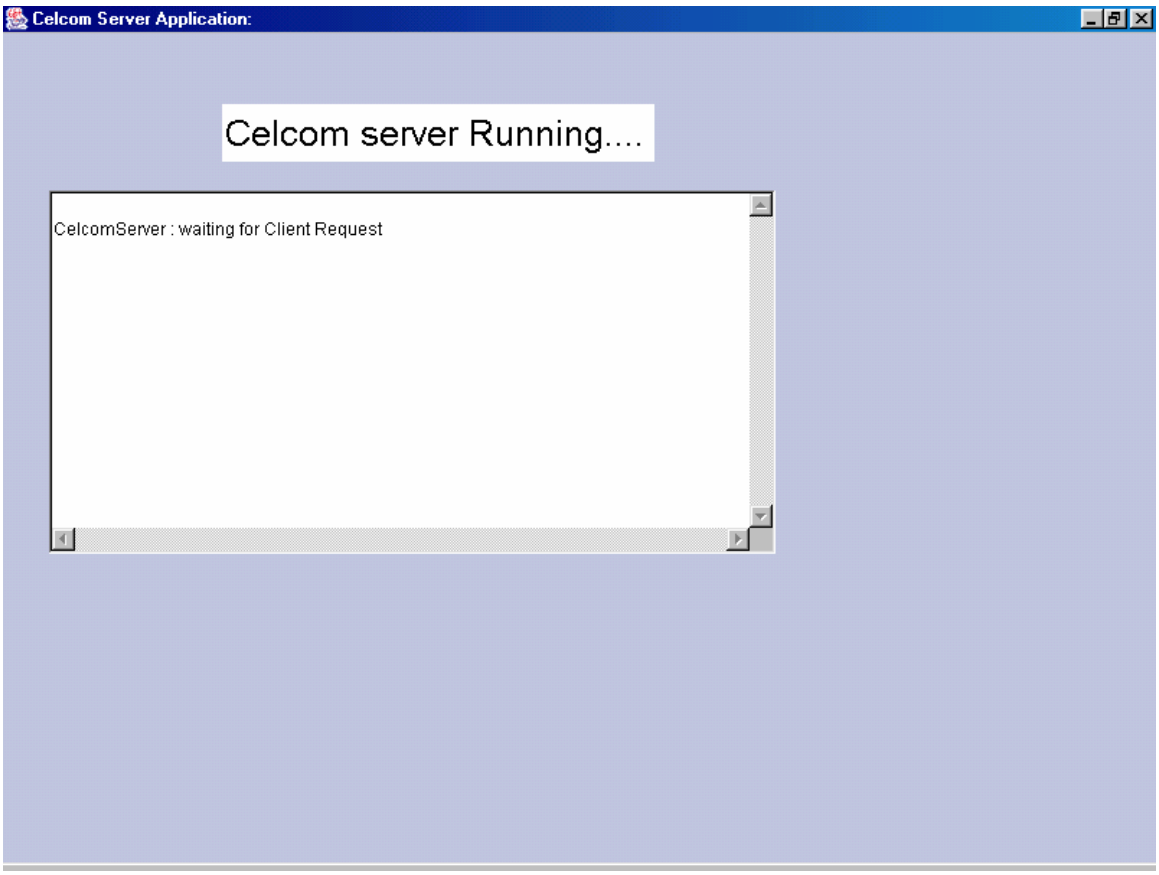


Fig 2. Shows Celcom Server Running.

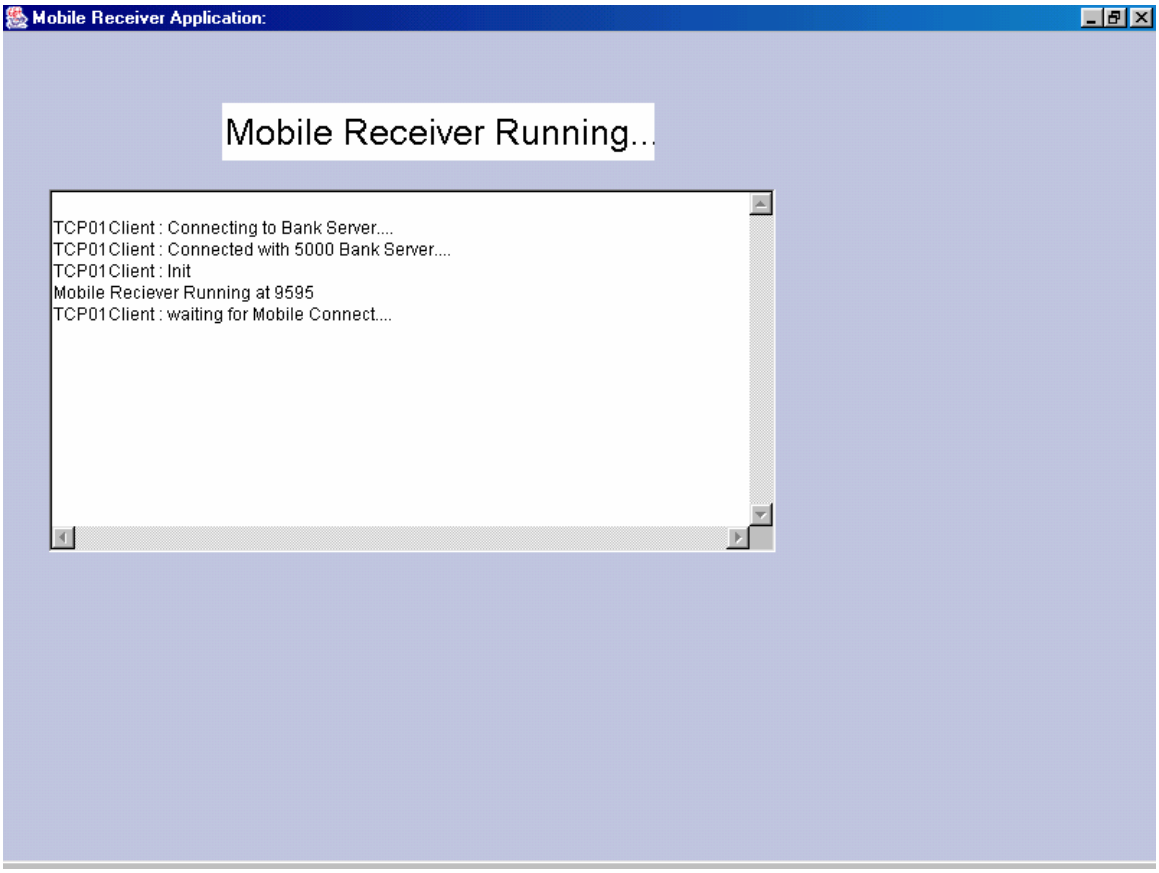


Fig 3. Shows Mobile server Running.

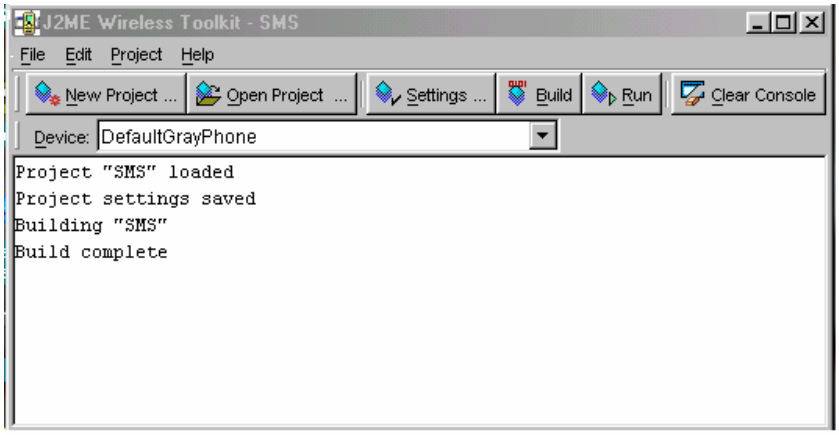


Fig 4. Shows the J2ME wireless toolkit usage.

Fig 5.Shows the options for SMS type.

Fig 6.Shows the screen for sending SMS regarding Recharging.

Fig 7. Screen after Recharging the mobile.

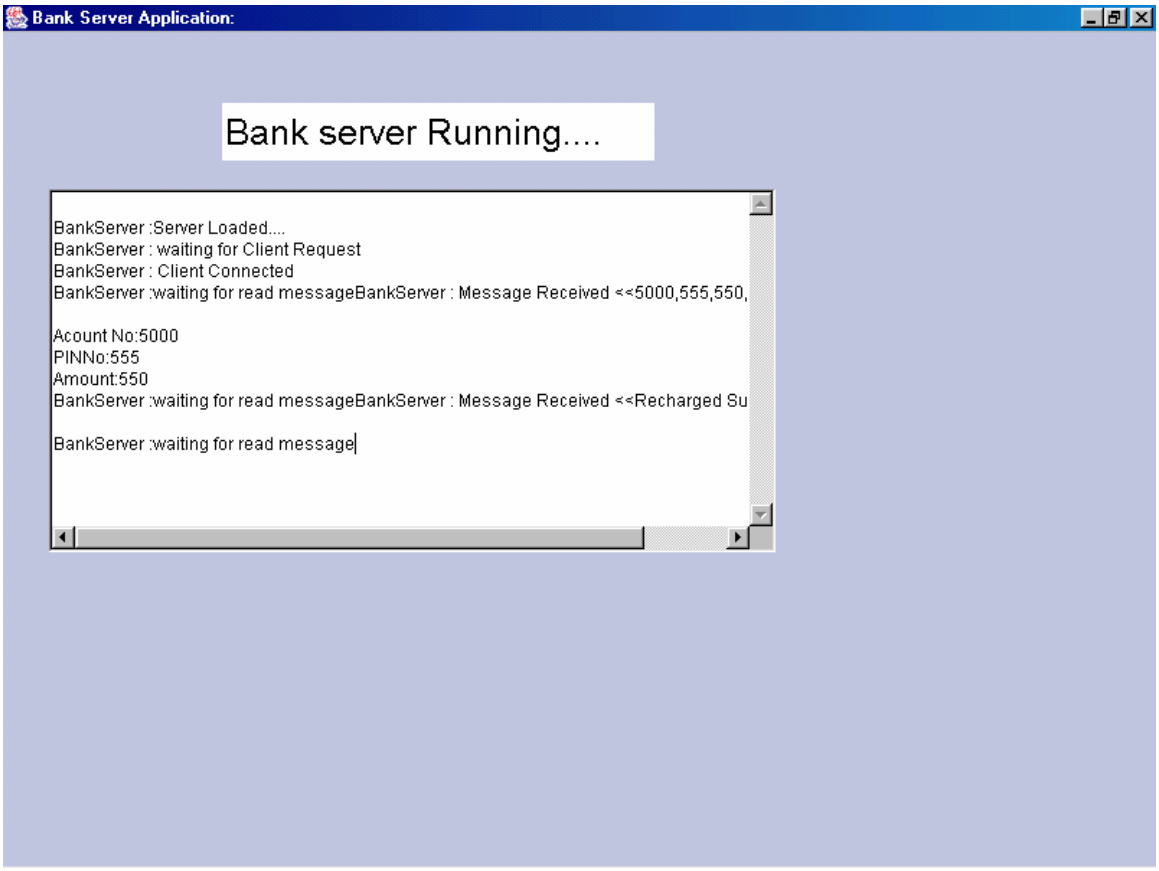


Fig 8. Shows the Bank server after updation of values with the message received.

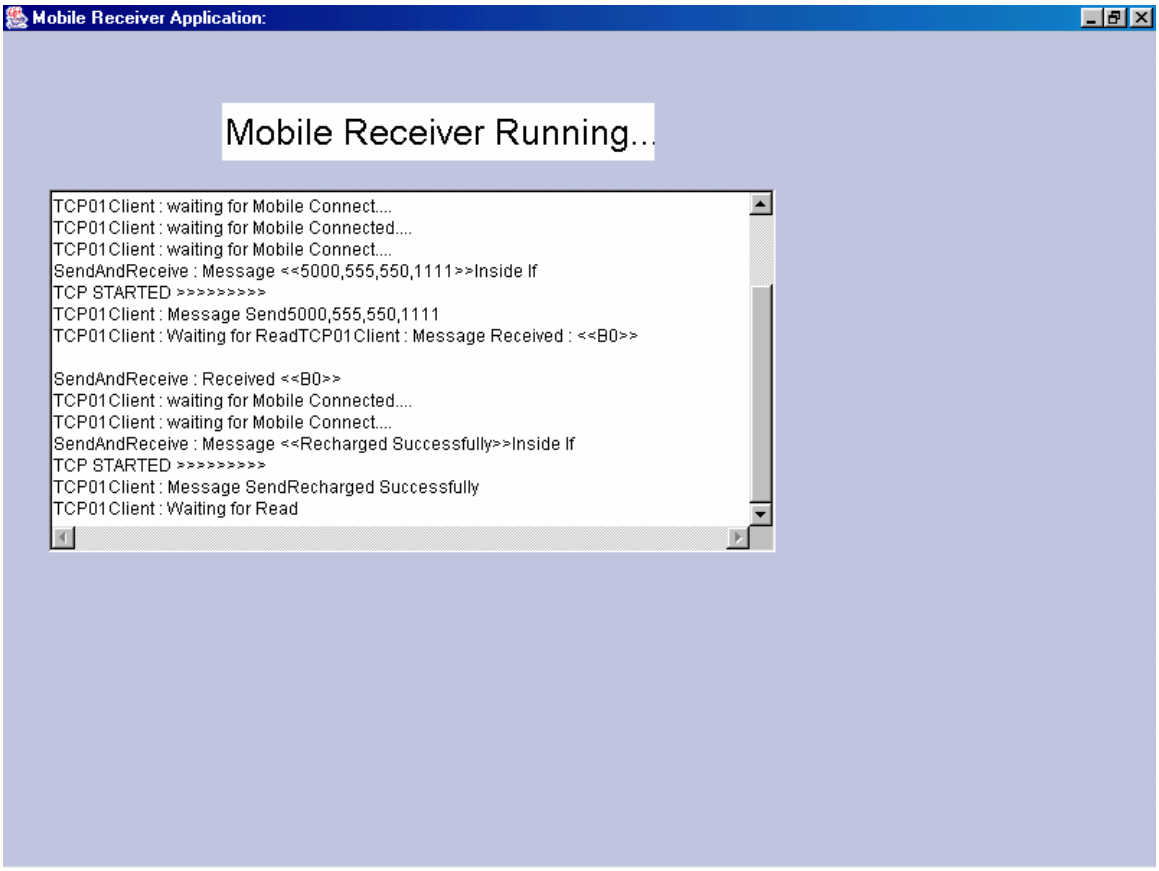


Fig 9. Mobile server after receiving the SMS from the client

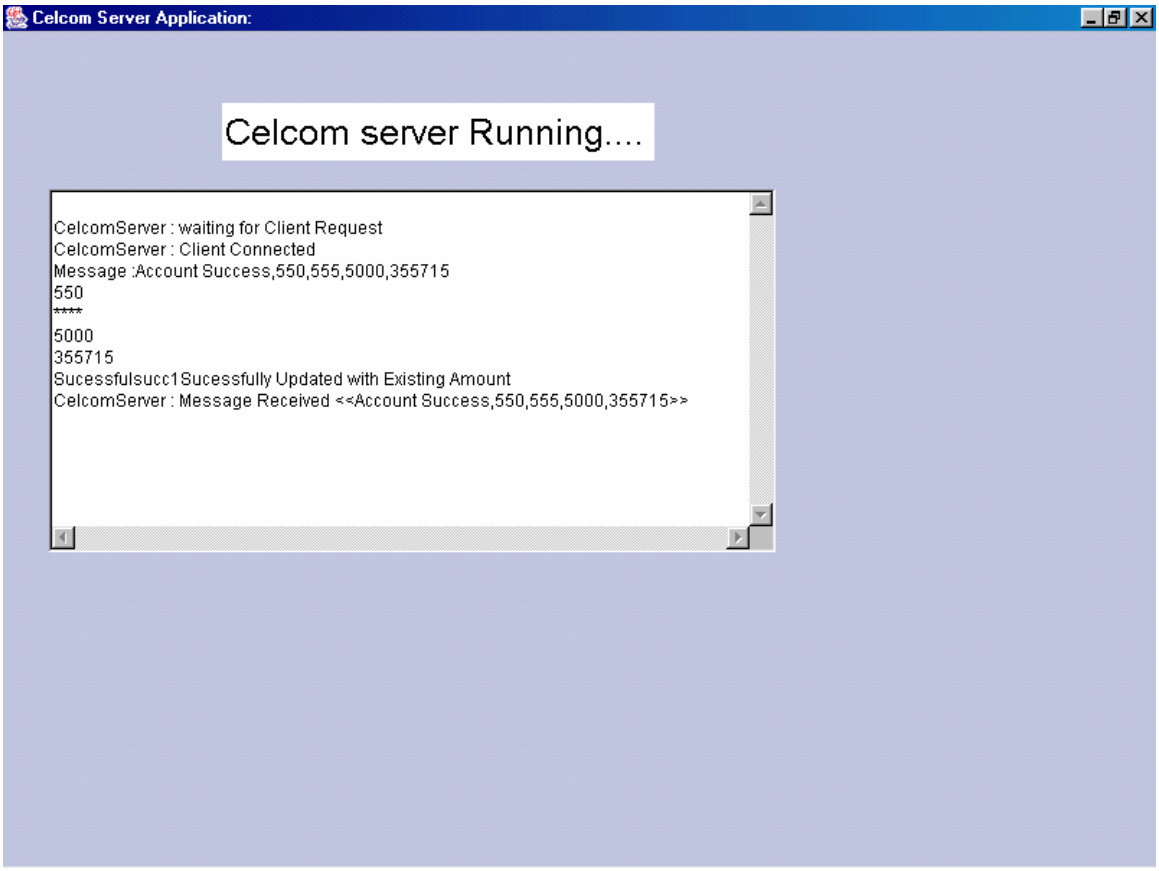


Fig 10. Celcom server sending message to mobile " Recharged successfully"

Fig 11. Shows the bank transaction screen

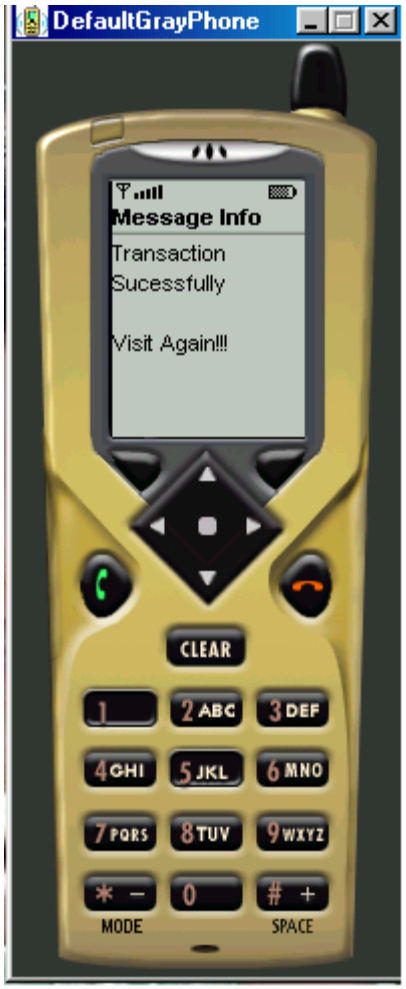


Fig 12. Shows mobile after Bank Transaction

Fig13. Shows the celcom server the message after Bank transaction

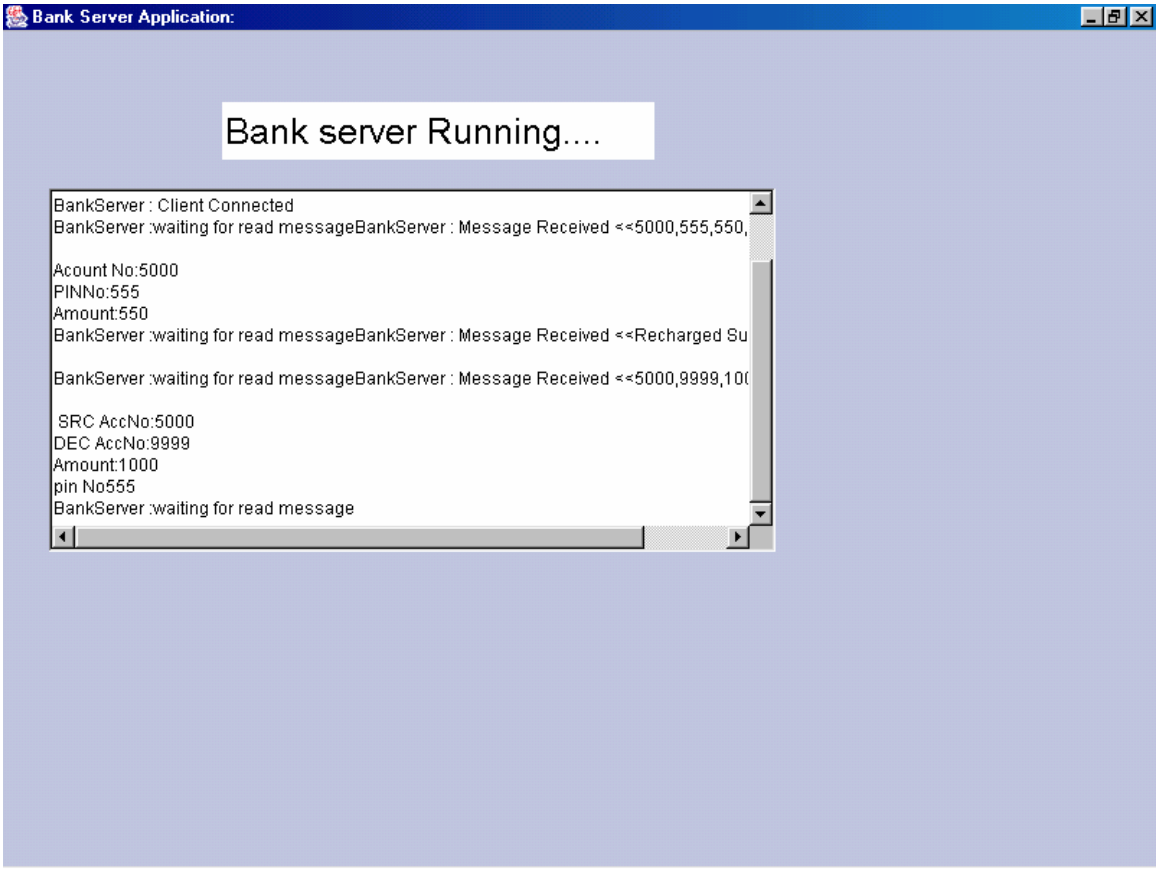


Fig 14. Shows the Bank server verifying the source and destination account

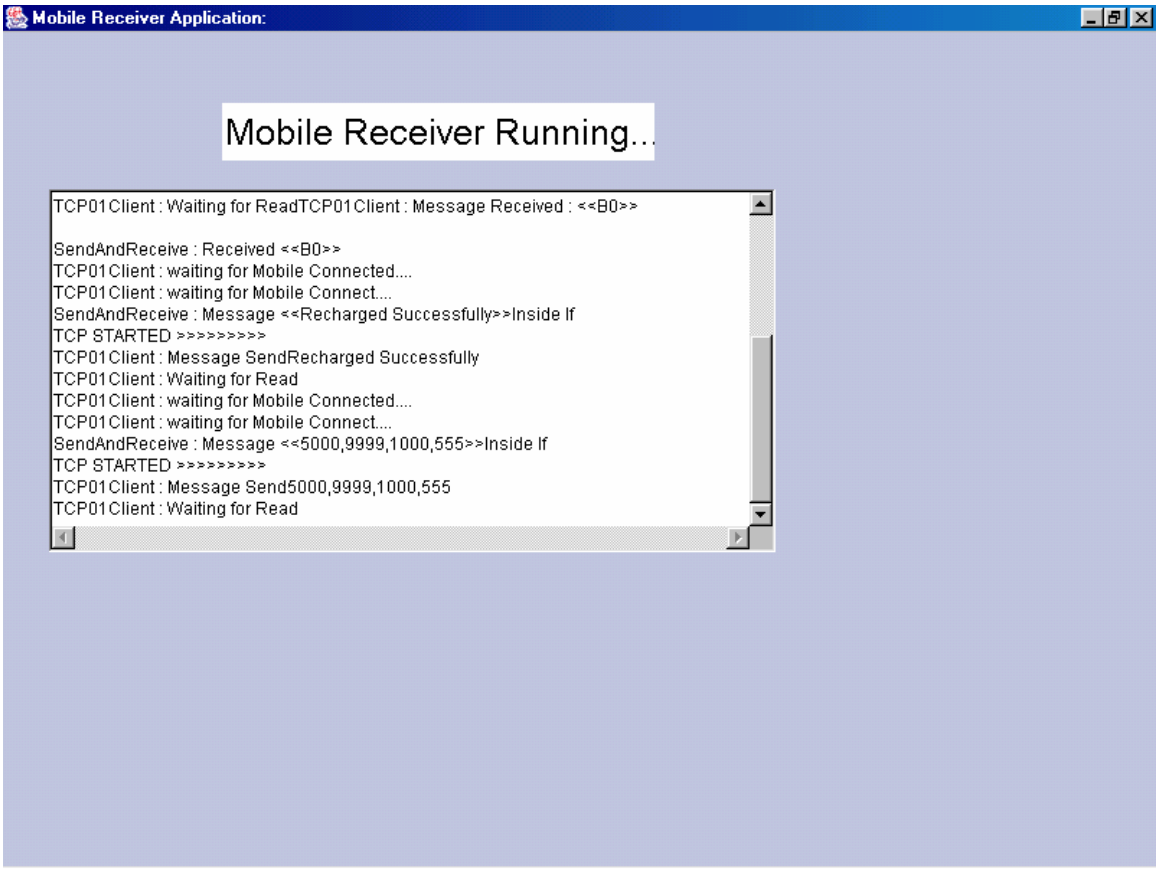


Fig 15. Mobile server Showing message received from the mobile

10.3 BIBLIOGRAPHY

- The Simple Book – By Marshal T. Rose
- Java 2 Tutorial
- J2ME Wireless Tool kit 4.1
- Windows 2000 help
- List of web sites
    - www.java.sun.com
    - www.ieng.com
    - www.rad.co.il
    - www.ibr.cs.tu-bs.de