Chapter 6

Implementation details of the system.

6.1. Introduction.

In the previous chapter it has been discussed about the analysis and design part of the system. Various design diagrams has been developed to identify the system. System design architecture has been developed in the analysis and design phase. In this chapter it is going to explain about the implementation details of each module. User interfaces are defined in the Appendix B.

6.2. Peer to Peer Module

Peer to Peer module contains sub modules like bank application module, SMS module, reporting module and database module. (See Appendix B to pseudo codes, coding parts etc)

SMS module used to send and receive SMS.

Software: - Java 5, JSMEngine API,

Hardware: - 3G modem, Computer.

Classes:- ConnectToModermServer

ReadMessageNew

SendMessageNew

StoreDBData

SysInfo

Code segments for above classes.

ConnectToModermServer.java

Code Segments

```
public static CService connectToServer(){
  try{
   comName = sysinfo.getSysInfo("name");
   comPort = sysinfo.getSysInfo("port");
   srv = new CService(comName, Integer.parseInt(comPort));
   //    Initialize service.
    srv.initialize();
    //   Set the cache directory.
    srv.setCacheDir(".\\");
    System.out.println("SERVER Connection Success");
  }catch(Exception e){
    System.out.println("]SendMessageNew()][connectToServer] "+e.toString());
  }
  return srv; }
```


In above code segment connection part to the modem has been developed. First it creates the object of the service and assign com port and baud rate as arguments to the constructor.

Above com port and baud rate values has been red from the configuration file. It is possible to change port and baud rate values without recompiling the code. This configuration file located in CD, AutomatedMoneyTransfer\SysConfig\ info.properties.

SysInfo.java class has been used to read this configuration information.

Eg:-

```
Properties props = new Properties();
props.load(new FileInputStream("SysConfig/info.properties"));
message = props.getProperty(propName);
```

Following code segment explain how to read message from the modem.

ReadMessageNew.java

```
status = srv.connect();
System.out.println("Connectio ok To read message " + status);
```

```
 if (status == CService.ERR_OK) {
  //Set the operation mode to PDU
  srv.setOperationMode(CService.MODE_PDU);
 if (srv.readMessages(msgList, CIncomingMessage.CLASS_REC_UNREAD) ==
CService.ERR_OK) {
 for (int i = 0; i < msgList.size(); i++) {
  CIncomingMessage msg = (CIncomingMessage) msgList.get(i);
  String Msisdn = msg.getOriginator();
  String newMsisdn = Msisdn.substring(3, Msisdn.length());  }}
```

ConnectToModermServer.java class create a service and using “srv.connect();” part of the above code connect to the device. Then get the status, if status is OK read the unread incoming messages. (CIncomingMessage.CLASS_REC_UNREAD).

SendMessageNew.java

This class used to send messages.

```
status = srv.connect();
srv.setOperationMode(CService.MODE_PDU);
srv.setSmscNumber("");
vecsms          = entity.sendPeerToPeerReplySMS();
 int ireplyVec   = vecsms.size();
try {
 while (j < ireplyVec) {
 record = vecsms.elementAt(j).toString();
strArr = record.split("\\|");
COutgoingMessage msg = new COutgoingMessage(strArr[0], strArr[1]);
 if (srv.sendMessage(msg) == CService.ERR_OK) {
    System.out.println("Reply Message Sent!" + strArr[0]);
     entity.updateReplyMsgSendSender(strArr[0]);
     entity.updateReplyMsgSendReceiver(strArr[0]);
 } else {
      System.out.println("Reply Message Failed!");
 } j++;}
```

Application connects to the server and get the sender list from the database iterates through the vector.

Then create object of “COutgoingMessage” class passing mobile number and message to be sent. So if the status is ok call “srv.sendMessage(msg)” method and send the sms.

(See Appendix B to pseudo codes).

6.3 Bank application module

Java network programming is used to implement this module.

Server socket and client socket applications are used.

Classes: - MobileOperatorClient, MobileOperatorServer

SenderToBank, StoreDBData

MobileOperatorClient.java

```
socket = new Socket(SERVER_HOSTNAME, SERVER_PORT);
is       = new DataInputStream(socket.getInputStream());
os       =  new PrintStream(socket.getOutputStream());
SenderToBank sender = new SenderToBank(os);
sender.setDaemon(true);
sender.start()
while ((message=is.readLine()) != null) {}
```


In above code segment it has been created a socket using bank server IP and port. Separate thread (SenderToBank) has been started to send transaction to the server. Using readLine() method it capture the reply from the server.

MobileOperatorServer.java

```
clientSocket = connection;
clientSocket.setSoTimeout(600000);
if (clientSocket.isConnected()) {
        out = new PrintWriter(clientSocket.getOutputStream(), true);}
in = new BufferedReader(new InputStreamReader(
```

clientSocket.getInputStream()));

while ((inputLine = in.readLine()) != null) {

String[] strArray = inputLine.split("\\|");

}

Above code used to get the input stream from the client and to split that protocol String. *(See Appendix B to pseudo codes, other coding parts etc)*

6.4 Cashless purchasing module.

Sub modules like SMS module, reporting module and database module are common to the cashless purchasing module as well. Except those there is a module called web application module. (See Appendix B to pseudo codes, coding parts etc)

Software:-JSP/AJAX

Classes:-CheckCustomerValidity.jsp,

CustValidateSubmit.jsp, MainMenu.jsp, RetFullTransReport.jsp, ViewStatus.jsp

Entityretailer, Customer.java, RetReport.java

Entityretailer.java

This class has been developed to retrieve data from the database and inserts data to the database.

public void createCustomerTxn(Customer customer){

PreparedStatement chkCust=null;

try{

chkCust = getDBConnection().prepareStatement("INSERT INTO retalier_cust_txn (transid,retailerID,custNIC,custMobile,amount,txndate,flag,cust_confir) values (?,?,?,?,?,?,?,?)");

chkCust.setString(1, createTransID());

chkCust.setString(2, customer.getRetailerID());

CheckCustomerValidity.jsp

```
function checkCustomerRegistred(strURL){
        if (window.XMLHttpRequest)     // Object of the current windows
        {           xmlHTTPReq = new XMLHttpRequest();       // Firefox,
Safari, ...}
        else if (window.ActiveXObject)  { // ActiveX version
        xmlHTTPReq = new ActiveXObject("Microsoft.XMLHTTP");  // IE }
        }
    var mobile   = txtMobile.value;
    strURL+="?mobile="+mobile+"&nic="+nic+"&amount="+amount;
 xmlHTTPReq.open('POST', strURL, true);
xmlHTTPReq.onreadystatechange = retriveStatus;
xmlHTTPReq.send(null); }
function retriveStatus(AJAX){
if (xmlHTTPReq.readyState == 4) {
updatepage(xmlHTTPReq.responseText);}}
```


Above code sample explain send XMLHTTPRequest and get the response (xmlHTTPReq.responseText) using AJAX.

6.5 Reporting Modules

Jasper IReport tool used to develop reports. Report structure is created using IReport. Intermediate XML file is created and this file used as an input to the java program. (See Appendix B to pseudo codes, coding parts etc)

Software:- Ireporting tool.

Jasper jar files.

Classes:-RetReport.java, PeerTransRptJ.java

txnRpt.jrxml

```
 <?xml version="1.0" encoding="UTF-8"  ?>
<!-- Created with iReport - A designer for JasperReports -->
```

<!DOCTYPE jasperReport PUBLIC "//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport
Above mentioned file is created when design the report using IReport tool.

PeerTransRptJ.java

This class is used to generate report using the jrxml file.

JasperReport jasperReport = JasperCompileManager.compileReport(reportSource);

Class.forName("com.mysql.jdbc.Driver").newInstance();

Connection conn = (Connectioin) DriverManager.getConnection(("jdbc:mysql://localhost :3306/moneytransfer',"root","root");

JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, params, conn);

JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, params, conn);

JasperExportManager.exportReportToHtmlFile(jasperPrint, reportDesc);

JasperViewer.viewReport(jasperPrint, false);

Above code sample describes how to run jasper xml file using the javacode. First compile the xml file and created report object. Then database connection created and fills report using data get from the database. Finally export report to the HTML file and view it. (*See Appendix B to other code samples and pseudo codes).*

6.6 Standard messages received in different status.

There are standard set of messages user will receive.

6.6.1 Peer to Peer money transfer state

When money receiver received the following message in success state.

Eg:-

"You have received 1000 of m-Cash from number <mobile number> on <date>. Auth key is <authkey>".

When money sender received following message in success state.

“You have credited <mobileno> with <amount> of m-Cash on <date>. Auth key is <authkey>”.

Money sender receives following message when sender does not have enough money to do the transaction.

“Sorry you have not sufficient balance to do this transaction”.

If transaction ID or other data can’t be validated following message will receive.

“Sorry your transaction can’t be validated. Please try with correct data”.

6.6.2 Cashless purchasing at super market.

Customer comes to the counter and asks to pay using m-Cash system. Then retailer checks the customer validity. If customer is a valid customer and customer is having sufficient balance, system will generate following message.

“Customer is registered with the facility”

If customer does not have enough money following message will display.

“Customer is not having enough balance to do the transaction”

If the customer is not registered with the facility following message will display.

“Customer is not registered with this facility”.

If customer is successfully registered, he/she will receive the following message.

<Merchant name> request to deduct <MMM.MM> amount of M-Cash from your account. Reply YES if you want to confirm. Reply NO if you don’t allow to do this transaction.

If customer reply YES following messages will receive both customer and retailer

“The owner of <mobile no> has been deducted by <amount> of M-Cash on dd/mm/yyyy and credit to the retailer account.” Retailer will receive the following message

“Credited Rs.<amount> to your account from <mobileno>”.

6.7 Package structure

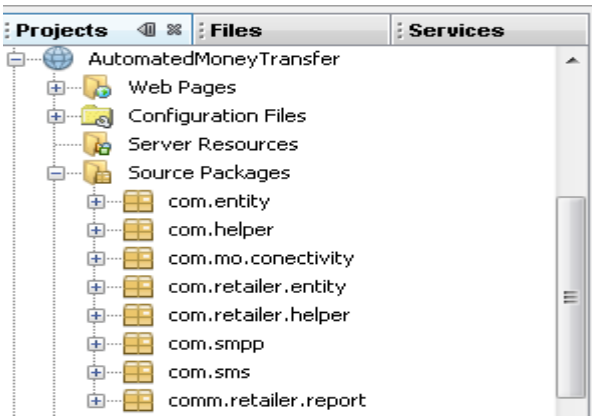

Figure 6.1 describes the package structure of the system.

6.8 Interface of sending SMS messages.

```
Inside send Message.
connection ok.0
Inside  While peer send
Reply Message Failed!
Inside  While peer send
Reply Message Sent!0714800123
Inside  While peer send
Reply Message Sent!0714812108
```

Figure 6.2 – SendMessageNew class interface.

6.9 Web interface for the Cashless purchasing system.

Below mentioned web interface is used by retailer. *(See Appendix B for other interfaces).*

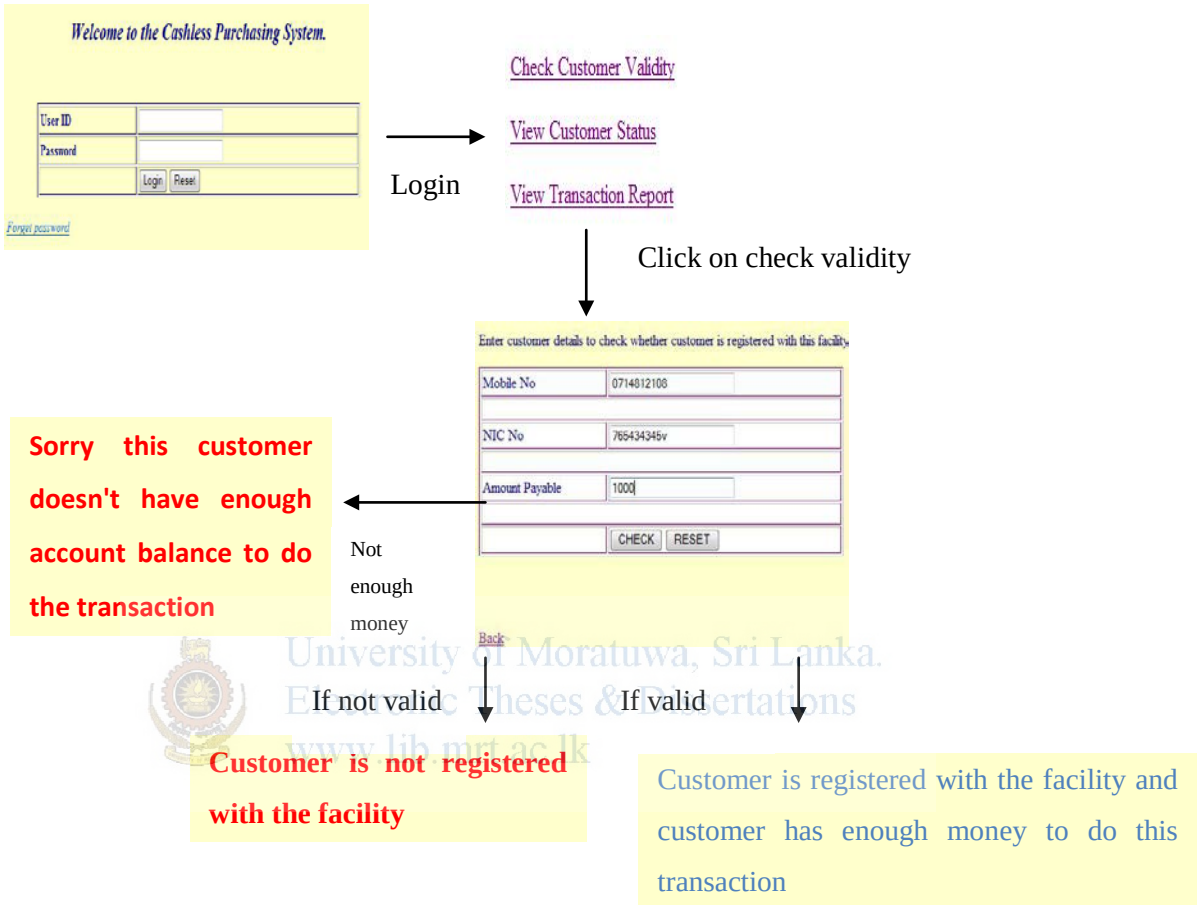

Figure 6.3 - Web interface for the Cashless purchasing system.

6.10 Summary

Chapter 6 describes the implementation details of the system. It contains sub sections like software used; pseudo codes, code sections, important message send and received by users and one example web user interfaces. Further user interfaces are included in the Appendix B and the system user guide.

Next chapter will describe about the evaluation of the project like how stable the system, achieve objectives, performance etc.