P- 1652

RECOVERY IN MOBILE WIRELESS ENVIRONMENT USING MOBILE AGENTS

A PROJECT REPORT

Submitted by

UMA. N

SAKTHI SANGARI. S

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY:: CHENNAI 600 025

MAY 2006

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**RECOVERY IN MOBILE WIRELESS ENVIRONMENT USING MOBILE AGENTS**" is the bonafide work of

"**Ms. UMA. N - 71202205053** and

Ms. SAKTHI SANGARI. S - 71202205062"

who carried out the project work under my supervision.

SIGNATURE

Dr.Gopalsamy. G

HEAD OF THE DEPARTMENT

Dept of Information Technology,

Kumaraguru College of Technology,

Coimbatore–641006.

SIGNATURE

Mrs.J.Cynthia M.E

SUPERVISOR

Senior Lecturer

Dept of Information Technology,

Kumaraguru College of Technology,

Coimbatore–641006.

The candidates with University Register Nos. 71202205053, 71202205062, were examined by us in the project viva-voce, examination held on 27/4/2006.

INTERNAL EXAMINER

27/4/06

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our sincere thanks to Dr.K.Arumugam, B.E. (Hons), and M.S. (USA), MIE, Correspondent, Kumaraguru College of Technology and the management, for providing us this opportunity to undertake this project work.

We express our profound gratitude to Dr.K.K.Padmanabhan B.Sc. (Engg), M.Tech., Ph.D., Principal, Kumaraguru College of Technology, Coimbatore, for permitting us to undergo a project work.

We are greatly indebted to Dr. Gopalsamy, Ph.D., Professor and Head of the Department of Information Technology, for the immense support he has provided us throughout our project.

We extend our sincere thanks to our project coordinator Prof.K.R.Baskaran, B.E., M.S., Assistant Professor, Department of Information Technology, for his constant support and encouragement.

We are much obliged to express our sincere thanks and gratitude to our beloved guide Mrs. J. Cynthia M.E, Department of Information Technology, for her valuable suggestions, constructive criticisms and encouragement which has enabled us to complete the project successfully.

We gratefully thank our Class Advisor Mrs. N.Suganthi M.E, Department of Information Technology, for extending her most appreciative and timely help to us.

We also thank all the teaching and non-teaching staff members of the Department of Information Technology for all their encouragement and moral support.

DECLARATION

We,

Uma. N **71202205053**

Sakthi Sangari. S **71202205062**

Declare that the project entitled **"Recovery in Mobile Wireless Environment Using Mobile Agents"**, submitted in partial fulfillment to Anna University as the project work of Bachelor Of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of Mrs.J.Cynthia M.E, Senior lecturer, Department of Information Technology, Kumaraguru College Of Technology, Coimbatore.

Place: Coimbatore
Date: 27/4/2006

N. Uma
[Uma. N]

S. Sakthi
[Sakthi Sangari. S]

Project Guided by

J. Cynthia

[Mrs. J. Cynthia M.E]

Dr. G. Gopalsamy,
Head of the Department,
Department of Information Technology
Kumaraguru College of Technology

ABSTRACT

Two technological advances in recent years have radically altered the nature of computing for most computer users. First is the mobility. Second advancement is the widespread use of internet. This has made possible the concept of mobile computing in which the user may access all their applications from any location. The mobility of these units makes it tricky to store application log and access it for recovery.

The system we have proposed is log management system for application recovery which uses a mobile-agent-based framework to facilitate seamless logging of application activities for recovery from transaction or system failure.

Application recovery is relatively more complex than database recovery because there are a large numbers of applications required to manage database processing. The objective is to facilitate seamless logging of application activities for recovery and minimize application recovery time.

Keywords: Mobile Database System (MDS), mobile agents, Global System for Mobile (GSM), coordinators, log unification, recovery

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 EXISTING SYSTEM AND ITS LIMITATIONS

In the existing system the complexity lies in:

1. managing database processing
2. the presence of multiple application states
3. the existence of random handoffs
4. the presence of operations in connected, disconnected and intermittent connection modes
5. unique processing demands of mobile units

Furthermore, it is not possible to store the entire log reliably at one location and retrieve it efficiently which makes it very difficult to "see" the entire log for recovery. The other problem in using mobile IP is *triangular routing* where all messages from the database server (DB) to the mobile unit (MU) have to be routed through the home agent. This invariably impedes application execution.

1.2 PROPOSED SYSTEM AND ITS ADVANTAGES

The mobile-agent based framework provides a platform for implementing recovery scheme based on the distributed logging approach which stores, retrieves, and unify fragments of application log for recovery. The system also eliminates the triangular routing problem and considers the case where a MU recovers in a BS different than the one in which it crashed.

Some of the benefits of the proposed system are:

1. reduces recovery time
2. scalable
3. minimizes wireless communication cost and time

1.3 MOBILE AGENT AND ITS ADVANTAGES

A mobile-agent is an autonomous program that can move from machine to machine in heterogeneous network under its own control. It can suspend its execution at any point, transport itself to a new machine, and resume execution from the point it stopped execution. An agent carries both the code and the application state.

Advantages of mobile agents are:

1. Protocol encapsulation

   Mobile agents can incorporate their own protocols in their code instead of depending on the legacy code provided by the host.

2. Robustness and fault tolerance

   When failures are detected, host systems can easily dispatch agents to other hosts. This ability makes the agents fault-tolerant.

3. Asynchronous and autonomous execution

   Once the agents are dispatched from a host, they can make decisions independently and autonomously.

2. SYSTEM REQUIREMENT ANALYSIS

2.1 PROJECT PLAN

In the problem analysis phase, the current system was analyzed by studying the papers referred in the References section. The present GSM architecture was analyzed and used as a foundation for the proposed system. A clear understanding of the needs of the system was framed, leading to the actual specification.

In the design phase, UML diagrams, DFDs and other suitable design diagrams were drawn. The different modules of the system and the interaction between these modules to produce the desired functionality were identified. The user interface was also designed.

Coding was done using MATLAB 6.5 software. Coding was written as modules to enable maintenance and easy testing and, debugging. The coding was also made in a presentable way and comments were added to ensure easy understanding.

Testing is done to check if the requirements of the user are met. It involves in checking the functionality of each module as per design document.

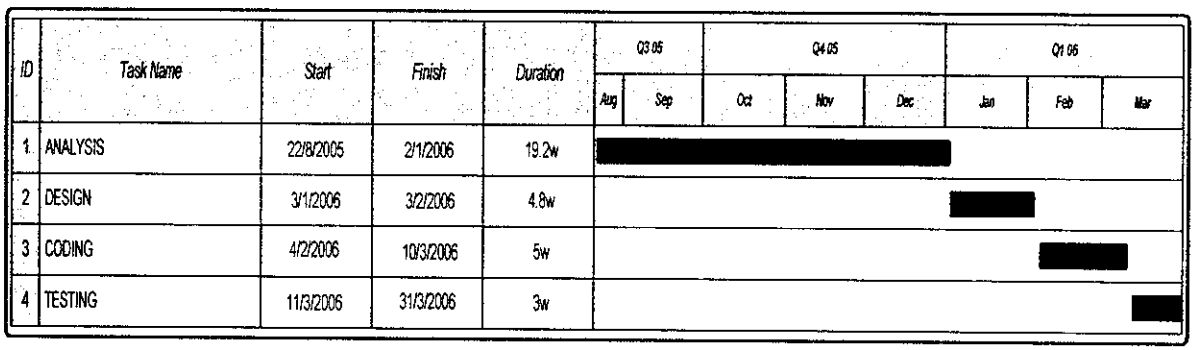

Figure 2.1 Project Schedule

2.2 SOFTWARE REQUIREMENT SPECIFICATION

2.2.1 PURPOSE

The purpose of this document is to specify the requirements of project "RECOVERY IN MOBILE WIRELESS ENVIRONMENT USING MOBILE AGENT". It describes the mobile agent framework for recovery from transaction or system failure. The goal is to reduce overall recovery time by efficiently managing resources and handoffs.

2.2.2 SCOPE

The scope of the document is, it describes application log management using a mobile-agent based framework is a GSM based mobile network.

2.2.3 FUNCTIONAL REQUIREMENTS

1. Validate ME user by checking the PIN code entered.
2. Check the access allowed for the MU by referring to the white list located in VLR.
3. Allow only one application to be loaded for every MU at any time.

4. Enable applications to connect to databases.
5. Backup application status and data for future recovery.

2.2.4 NON-FUNCTIONAL REQUIREMENTS

1. Provide a user-friendly user interface
2. Application response time should not exceed 2 seconds.
3. Provide help features to enable easy navigation.

2.2.5 DEVELOPMENT AND OPERATING ENVIRONMENT

2.2.5.1 Software Specifications

Operating System: Windows 2000/XP

Software: MATLAB 6.5

2.2.5.2 Hardware Specifications

Processor: Intel Pentium

RAM: 128 MB RAM minimum, 256 MB RAM recommended

2.2.6 DEFINITION

2.2.6.1 Customer

The customer is the ultimate recipient of the developed product and its artifact

2.2.6.2 User

A person who will use the developed system

2.2.6.3 Analyst

The analyst details the specification of the system's functionality by describing the requirements aspect and other supporting software requirements.

2.2.7 LIST OF ABBREVIATION

AUC –Authentication Center

BS – Base Station

DBS – Database Server

EIR – Equipment Identity Register

GSM – Global System for Mobile

HLR – Home Location Register

IMEI – International Mobile Equipment Identifier

IMSI – International Mobile Subscriber Identifier

ME – Mobile Equipment

MSC – Mobile Switching Center

MSISDN – Mobile Subscriber ISDN

MU – Mobile Unit

PSTN – Public Switched Telephone Network

VLR – Visitor Location Register

MSC. It is termed the VLR because it holds the information on those subscribers that are visiting its VLR area.The AUC is solely used to store information that is concerned with GSM security features i.e., user authentication and radio path encryption. The AUC will only ever communicate with the HLR.

The EIR is used to store three different lists of IMEI. The white list contains a series of IMEI that have been allocated to ME that may used on the GSM network. The black list contains IMEI of all MEs that must be barred from using the GSM network. The grey list holds the IMEI of MEs that must be tracked by the network for evaluation purposes.

The MSC is concerned with the routing of calls to and from the mobile users. It possesses a large switching capability that varies between manufacturers, but a typical MSC will control a few tens of BSCs. The MSC's similar to the switching exchange in a fixed network. The OMC provides the means by which the operator controls the network. Each OMC will typically be in charge of a subsystem, for example, the BSS.

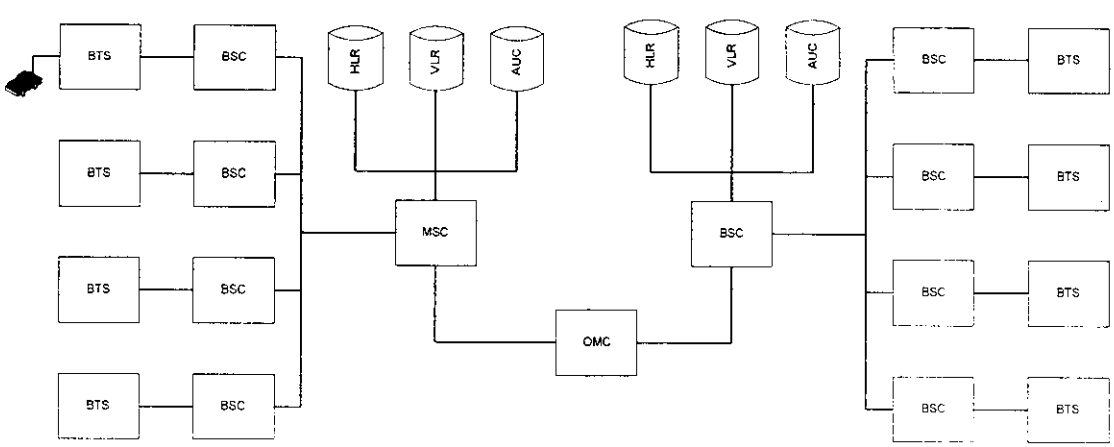

Figure 3.1 GSM Network Architecture

3.2 GSM-BASED MOBILE APPLICATIONS

3.2.1 FLIGHT TICKET RESERVATION

1. Increase sales through improved customer service.
2. Increase safety as a result of improved recording of maintenance data.

3.2.2 FUNDS TRANSFER

1. Reduce need for visits to the bank and instead spend more time with customer
2. Eliminate or reduce references to paperwork

3.2.3 WEATHER REPORT

1. View instant weather report

3.3 MATLAB – INTRODUCTION

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

1. Math and computation
2. Algorithm development
3. Modeling, simulation, and prototyping
4. Data analysis, exploration, and visualization
5. Scientific and engineering graphics

6. Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB uses software developed by the LAPACK and ARPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which

toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

3.4 MATLAB SYSTEM

The MATLAB system consists of five main parts:

1. Development Environment. This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.
2. The MATLAB Mathematical Function Library. This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen values, Bessel functions, and fast Fourier transforms.
3. The MATLAB Language. This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.
4. Handle Graphics. This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data

visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

5. The MATLAB Application Program Interface (API). This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

3.5 MATLAB – DEVELOPMENT ENVIRONMENT

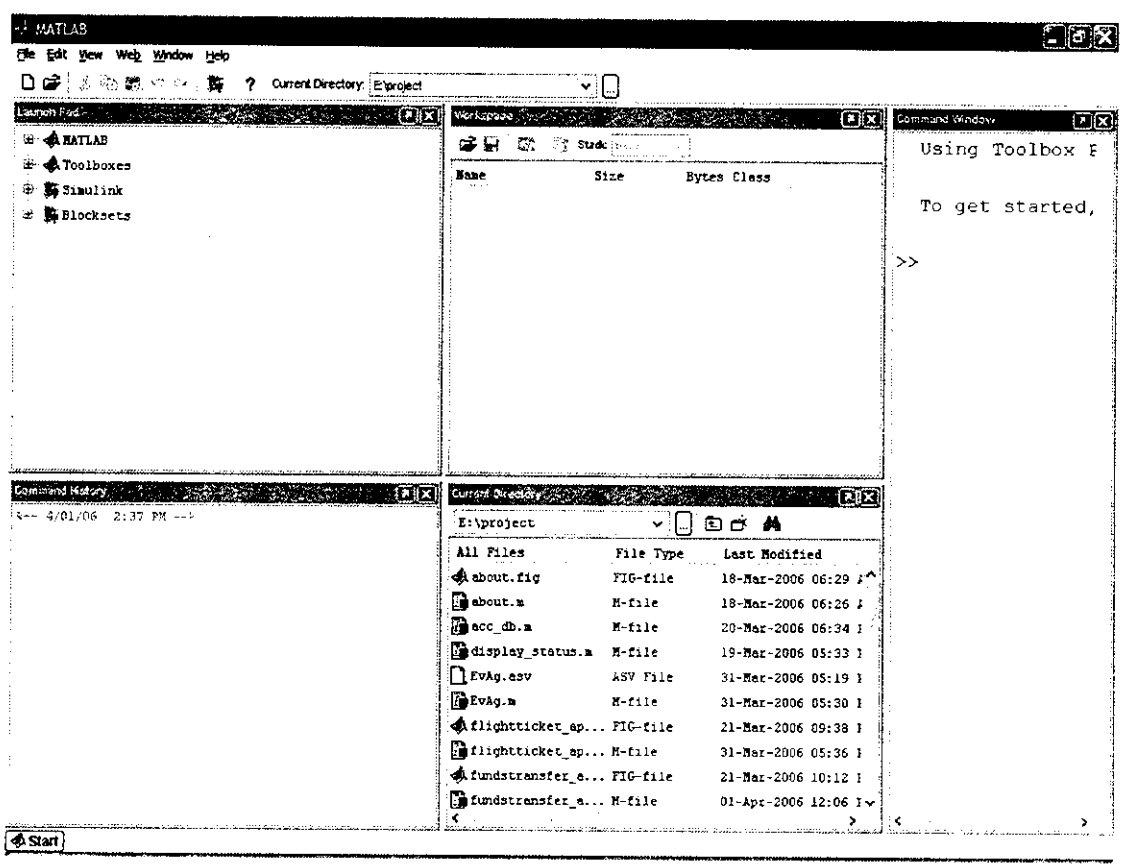

Figure 3.2 MATLAB – Development Environment

4. DESIGN DOCUMENTS

4.1 DATA FLOW DIAGRAMS

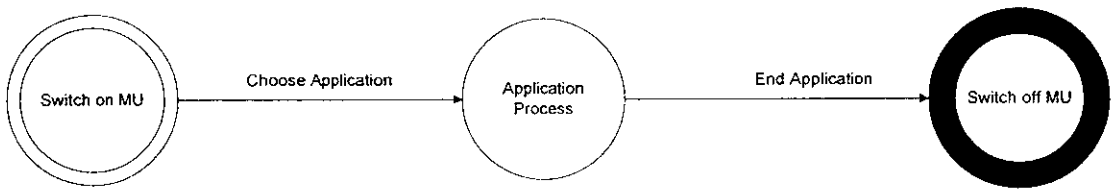

Figure 4.1 DFD Level 0

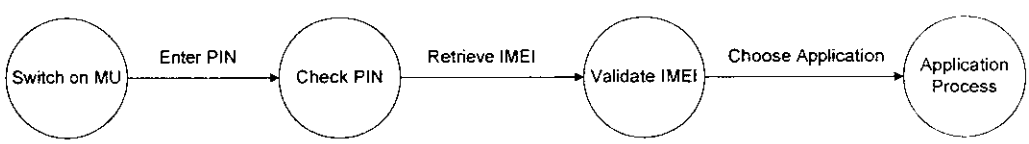

Figure 4.2 DFD Level 1

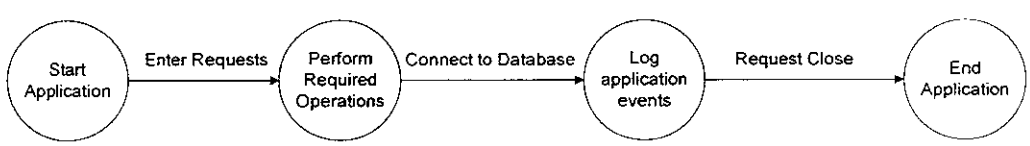

Figure 4.3 DFD Level 2

4.2 INPUT DESIGN

In our project, for the MU, the PIN number is given as input to validate it after it is switched on. There are three lists, namely, black list, grey list, white list. The BS retrieves the IMEI from the validated MU and sends it to the MSC. The MSC checks in which list the IMEI is present.

There are three sample applications in our project for the MU. The first application is Funds Transfer which is a banking application. The user must

enter the Login ID and password to logon to the application. The Login ID is associated to the user's account number. In order to transfer funds from one account to another the user needs to enter the appropriate details such as the recipient's account number and amount.

The second application is Flight Ticket Reservation. Here the inputs to be given are flight name, class, date, number of seats, source and the destination to reserve tickets. The third application is Weather Report. Here on selecting the place, the climatic conditions of that place can be viewed.

4.3 PROCESS DESIGN

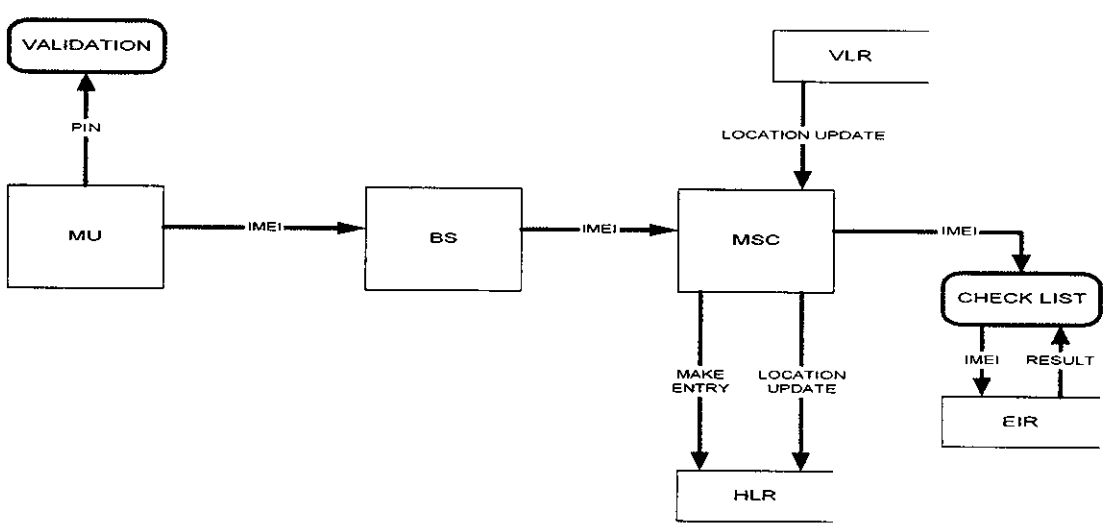

Figure 4.4 Validation Process

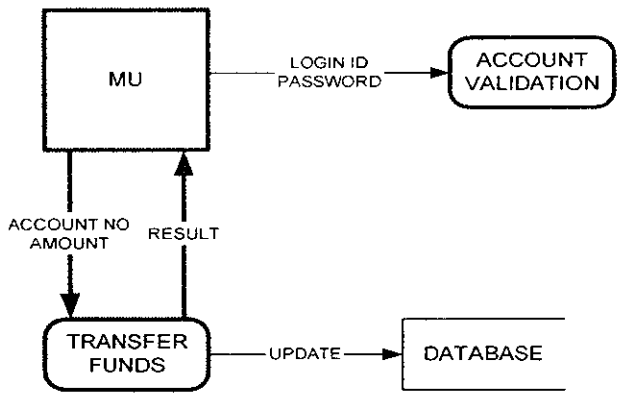

Figure 4.5 Funds Transfer Application

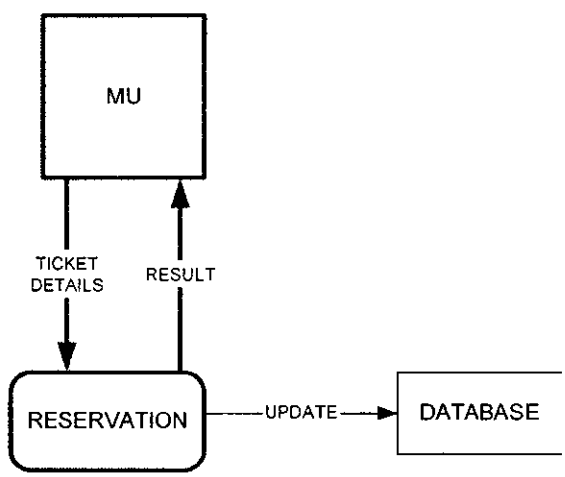

Figure 4.6 Flight Tickets Reservation Application

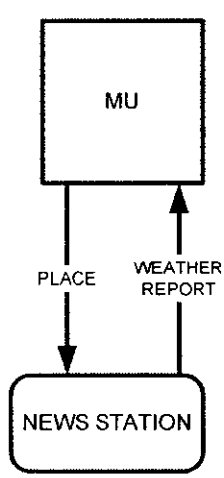

Figure 4.7 Weather Report Application

4.4 OUTPUT DESIGN

In the output design, the application is designed to show the MU's MSISDN and its status. In the Funds transfer application, on entering the transaction details and clicking the Submit button, a message box appears indicating whether the transaction is successful or not. In the Flight Ticket Reservation application, on entering the details and clicking the OK button, a message box appears indicating that the reservation is made. In the Weather Report application the climatic conditions are displayed.

For each application, a log is prepared. The log is identified using the MU's MSISDN. The log contains the following details:

1. date
2. time
3. VLR location
4. application events

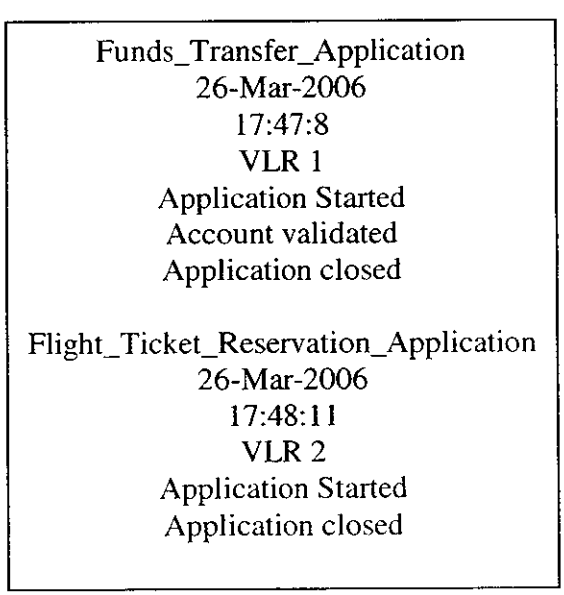

Figure 4.8 Sample Log

4.5 DATABASE DESIGN

HLR TABLE

PK	IMSI	Varchar(15)
PK	MSISDN	Varchar(15)
	LOCATION	Varchar(10)
	STATUS	Varchar(10)

Table 4.1 Details of HLR

VLR TABLE

PK	IMSI	Varchar(15)
PK	MSISDN	Varchar(15)
	HLR	Varchar(10)

Table 4.2 Details of VLR

EIR TABLE

PK	IMEI	Varchar(15)
	STATUS	Varchar(10)

Table 4.3 Details of EIR

FUNDS TRANSFER TABLE

PK	LOGINID	Varchar(10)
PK	ACCOUNT NUMBER	Varchar(10)
	PASSWORD	Varchar(10)
	BALANCE	Number(10,2)

Table 4.4 Details of Funds Transfer

FLIGHT TICKET RESERVATION TABLE

PK	SNO	AutoNumber
	FLIGHT NAME	Varchar(10)
	CLASS	Varchar(10)
	DATE	Date
	NO OF SEATS	Number(5)
	SOURCE	Varchar(20)
	DESTINATION	Varchar(20)

Table 4.5 Details of Flight Ticket Reservation

5. PRODUCT TESTING

Testing is done to detect the errors in the software. This implies not only to the coding phase but to uncover errors introduced in all the previous phases. The following are the types of tests that were performed.

5.1 UNIT TESTING

Each and every module is tested separately to check if its intended functionality is met. Some unit testing performed are:

1. Validating the MU
2. Loading Database and applications
3. Performing log for applications

5.2 INTEGRATION TESTING

It is the testing performed to detect errors on interconnection between modules. The MU once switched on and validated, should allow the loading of applications. The applications should connect to respective databases. The application events should be backed up in the log files for future recovery.

5.3 SYSTEM TESTING

The system is tested against the system requirements to see if all the requirements are met and if the system performs as per the specified requirements. The system is tested as a whole to check for its functionality.

5.4 VALIDATION TESTING

This test is done to check for the validity of the entered input. The user inputs to the corresponding application input fields are verified before updating in the database.

6. FUTURE ENHANCEMENTS

In our project Mobile Banking has been deployed using mobile applications that have been developed as Standalone Mobile Application Clients. In future, this can be enhanced by using Wireless Application Protocol (WAP) and Interactive Voice Response (IVR) to enable advanced Internet and office applications.

Our project could be further developed into a robust and highly available mobile information management system which is the backbone of e-commerce and m-commerce platforms.

Only experience can give us more insight into this vast research field and allow us to develop more efficient solutions to these and various other problems that we come across in such highly unreliable and complicated systems.

7. CONCLUSION

In this project, we presented a mobile-agent-based framework for supporting application recovery in a mobile wireless environment. The distributed logging scheme implemented in our project improves the recovery time in situations where the failure time is nontrivial by storing, retrieving and unifying fragments of the application log for recovery. Our framework supports more than one application and logging scheme. Our scheme minimizes wireless communication cost and time.

8. APPENDIX

8.1 SAMPLE CODE

START_APPLN.M

```
function varargout = start_appln(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @start_appln_OpeningFcn, ...
                   'gui_OutputFcn',  @start_appln_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

function start_appln_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = start_appln_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
close(start_appln)
main_appln
```

MAIN_APPLN.M

```
function varargout = main_appln(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @main_appln_OpeningFcn, ...
                   'gui_OutputFcn',  @main_appln_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

function main_appln_OpeningFcn(hObject, eventdata, handles, varargin)
clc
global time1
global time2
global time3
global status1
global status2
global status3
global msisdnstr1
global msisdnstr2
global msisdnstr3
time1 = fix(cputime)-1;
time2 = fix(cputime)-1;
time3 = fix(cputime)-1;
retrieve_msisdn_imei;
status1 = 'Off';
```

```
status2 = 'Off';
status3 = 'Off';
handles.output = hObject;
guidata(hObject, handles);
set(handles.Mobile1, 'Value', 1);
set(handles.Mobile2, 'Value', 0);
set(handles.Mobile3, 'Value', 0);
display_status(handles,'off');
set(handles.label1, 'String', msisdnstr1);
set(handles.label2, 'String', msisdnstr2);
set(handles.label3, 'String', msisdnstr3);

function varargout = main_appln_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function File_Callback(hObject, eventdata, handles)

function Exit_Callback(hObject, eventdata, handles)
close

function Help_Callback(hObject, eventdata, handles)
help1;

function About_Callback(hObject, eventdata, handles)
about;

function status_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function status_Callback(hObject, eventdata, handles)
```

```
function Mobile1_Callback(hObject, eventdata, handles)
global status1
set(handles.Mobile1, 'Value', 1);
set(handles.Mobile2, 'Value', 0);
set(handles.Mobile3, 'Value', 0);
display_status(handles,status1);

function Mobile2_Callback(hObject, eventdata, handles)
global status2
set(handles.Mobile1, 'Value', 0);
set(handles.Mobile2, 'Value', 1);
set(handles.Mobile3, 'Value', 0);
display_status(handles,status2);

function Mobile3_Callback(hObject, eventdata, handles)
global status3
set(handles.Mobile1, 'Value', 0);
set(handles.Mobile2, 'Value', 0);
set(handles.Mobile3, 'Value', 1);
display_status(handles,status3);

function Start_Application_Callback(hObject, eventdata, handles)

function Funds_Transfer_Callback(hObject, eventdata, handles)
global time1
global time2
global time3
global status1
global status2
global status3
global handles_copy
if get(handles.Mobile1,'Value')
   switch status1
```

```
        case { 'validated', 'application closed' }
            handles_copy = handles;
            login(1);
            status1 = 'application loaded';
            display_status(handles,status1);
            EvAg(1);
            time1 = time1 + 3;
        otherwise
            warndlg('Not allowed');
    end
elseif get(handles.Mobile2,'Value')
    switch status2
        case { 'validated', 'application closed' }
            handles_copy = handles;
            login(2);
            status2 = 'application loaded';
            display_status(handles,status2);
            EvAg(2);
            time2 = time2 + 3;
        otherwise
            warndlg('Not allowed');
    end
elseif get(handles.Mobile3,'Value')
    switch status3
        case { 'validated', 'application closed' }
            handles_copy = handles;
            login(3);
            status3 = 'application loaded';
            display_status(handles,status3);
            EvAg(3);
            time3 = time3 + 3;
        otherwise
            warndlg('Not allowed');
    end
```

```
end


function Flight_Ticket_Reservation_Callback(hObject, eventdata, handles)
global time1
global time2
global time3
global status1
global status2
global status3
global handles_copy
if get(handles.Mobile1,'Value')
    switch status1
    case { 'validated', 'application closed' }
      handles_copy = handles;
      flightticket_appln(1);
      status1 = 'application loaded';
      display_status(handles,status1);
      EvAg(1);
      time1 = time1 + 3;
    otherwise
      warndlg('Not allowed');
  end
elseif get(handles.Mobile2,'Value')
  switch status2
    case { 'validated', 'application closed' }
      handles_copy = handles;
      flightticket_appln(2);
      status2 = 'application loaded';
      display_status(handles,status2);
      EvAg(2);
      time2 = time2 + 3;
    otherwise
      warndlg('Not allowed');
```

```
    end
elseif get(handles.Mobile3,'Value')
    switch status3
        case { 'validated', 'application closed' }
            handles_copy = handles;
            flightticket_appln(3);
            status3 = 'application loaded';
            display_status(handles,status3);
            EvAg(3);
            time3 = time3 + 3;
        otherwise
            warndlg('Not allowed');
    end
end

function Weather_Report_Callback(hObject, eventdata, handles)
global time1
global time2
global time3
global status1
global status2
global status3
global handles_copy
if get(handles.Mobile1,'Value')
    switch status1
        case { 'validated', 'application closed' }
            handles_copy = handles;
            weatherreport_appln(1);
            status1 = 'application loaded';
            display_status(handles,status1);
            EvAg(1);
            time1 = time1 + 3;
        otherwise
            warndlg('Not allowed');
```

```
    end
elseif get(handles.Mobile2,'Value')

    switch status2
        case { 'validated', 'application closed' }
            handles_copy = handles;
            weatherreport_appln(2);
            status2 = 'application loaded';
            display_status(handles,status2);
            EvAg(2);
            time2 = time2 + 3;
        otherwise
            warndlg('Not allowed');
    end
elseif get(handles.Mobile3,'Value')
    switch status3
        case { 'validated', 'application closed' }
            handles_copy = handles;
            weatherreport_appln(3);
            status3 = 'application loaded';
            display_status(handles,status3);
            EvAg(3);
            time3 = time3 + 3;
        otherwise
            warndlg('Not allowed');
    end
end

function Mobile_Unit_Options_Callback(hObject, eventdata, handles)

function Switch_Callback(hObject, eventdata, handles)
global time1
global time2
global time3
```

```
global status1
global status2
global status3
registers_database;
if get(handles.Mobile1,'Value')
   if strcmp(status1,'Off')
      status1 = 'On ';
      display_status(handles,status1);
      status1 = validate;
      if strcmp( status1, 'validated' )
         result1 = msc(1, 1);
         EvAg(1);
         time1 = time1 + 3;
      end
   else
      status1 = 'Off';
   end
   display_status(handles,status1);
elseif get(handles.Mobile2,'Value')
   if strcmp(status2,'Off')
      status2 = 'On ';
      display_status(handles,status2);
      status2 = validate;
      if strcmp( status2, 'validated' )
         result2 = msc(1, 2);
         EvAg(2);
         time2 = time2 + 3;
      end
   else
      status2 = 'Off';
   end
   display_status(handles,status2);
elseif get(handles.Mobile3,'Value')
   if strcmp(status3,'Off')
```

```
        status3 = 'On ';
        display_status(handles,status3);
        status3 = validate;
        if strcmp( status3, 'validated' )
           result3 = msc(1, 3);
           EvAg(3);
           time3 = time3 + 3;
        end
     else
        status3 = 'Off';
     end
     display_status(handles,status3);
end
```

RETRIEVE_MSISDN_IMEI.M

```
global msisdnstr1
global msisdnstr2
global msisdnstr3
global imeistr1
global imeistr2
global imeistr3
n = randint(1,3,[1,20]);
msisdn_samples(1) = 919894393659;
msisdn_samples(2) = 919894375185;
msisdn_samples(3) = 919894360748;
msisdn_samples(4) = 919894370254;
msisdn_samples(5) = 919894385844;
msisdn_samples(6) = 919894382153;
msisdn_samples(7) = 919894376112;
msisdn_samples(8) = 919894327265;
msisdn_samples(9) = 919894332261;
msisdn_samples(10) = 919894370655;
msisdn_samples(11) = 919894346412;
msisdn_samples(12) = 919894385776;
```

```
msisdn_samples(13) = 919894348514;
msisdn_samples(14) = 919894344677;
msisdn_samples(15) = 919894378457;
msisdn_samples(16) = 919894353355;
msisdn_samples(17) = 919894327265;
msisdn_samples(18) = 919894364268;
msisdn_samples(19) = 919894378657;
msisdn_samples(20) = 919894312356;
msisdnstr1 = int2str(msisdn_samples(n(1)));
msisdnstr2 = int2str(msisdn_samples(n(2)));
msisdnstr3 = int2str(msisdn_samples(n(3)));
imei_samples(1) = 353382001573463;
imei_samples(2) = 356289618234137;
imei_samples(3) = 383971113557651;
imei_samples(4) = 335776864644906;
imei_samples(5) = 332112297869664;
imei_samples(6) = 362387165931599;
imei_samples(7) = 322868548637678;
imei_samples(8) = 313970638677486;
imei_samples(9) = 374181718147721;
imei_samples(10) = 327342135755271;
imei_samples(11) = 334016547722241;
imei_samples(12) = 342566324778816;
imei_samples(13) = 331830833656257;
imei_samples(14) = 388423428816362;
imei_samples(15) = 345925695423852;
imei_samples(16) = 363215373443133;
imei_samples(17) = 358819584868792;
imei_samples(18) = 356438893912768;
imei_samples(19) = 354822505351298;
imei_samples(20) = 357382312238691;
imeistr1 = int2str(imei_samples(n(1)));
imeistr2 = int2str(imei_samples(n(2)));
imeistr3 = int2str(imei_samples(n(3)));
```

DISPLAY_STATUS.M

```
function display_status(handles,str)
set( handles.status, 'String', str );
return;
```

MSC.M

```
function result = msc(option, n)
switch option
   case 1
      result = track_imei(n);
      return;
   case 2
      result = loc_update(n);
      return;
end

function result = track_imei(n)
global hlr
global eir
global imeistr1
global imeistr2
global imeistr3
result = 't';
switch n
   case 1
      imeistr = imeistr1;
   case 2
      imeistr = imeistr2;
   case 3
      imeistr = imeistr3;
end
for i = 1:20
   if imeistr == eir(i).IMEI
```

```
      check = eir(i).STATUS;
      switch check
        case 1
          result = 't';
          return;
        case 2
          result = 'f';
          warndlg('Access Denied', 'Network Operator');
          return;
      end
    end
end

function result = loc_update(n)
global hlr
global vlr
global msisdnstr1
global msisdnstr2
global msisdnstr3
global conn
global curs1
switch n
   case 1
      msisdnstr = msisdnstr1;
   case 2
      msisdnstr = msisdnstr2;
   case 3
      msisdnstr = msisdnstr3;
end
   for i = 1:20
      if vlr(i).MSISDN == msisdnstr
        if vlr(i).ID == 'V1'
          vlr(i).ID = 'V2';
          exdata = cellstr(vlr(i).ID);
```

```
            where1 = 'where MSISDN = ''';
            where2 = msisdnstr;
            where3 = '''';
            whereclause = strcat(where1,where2,where3);
            update(conn, 'VLR', {'ID'}, exdata, whereclause);
            break;
        else
            vlr(i).ID = 'V1';
            exdata = cellstr(vlr(i).ID);
            where1 = 'where MSISDN = ''';
            where2 = msisdnstr;
            where3 = '''';
            whereclause = strcat(where1,where2,where3);
            update(conn, 'VLR', {'ID'}, exdata, whereclause);
            break;
        end
    end
end
for i = 1:20
    if hlr(i).MSISDN == msisdnstr
        if hlr(i).LOC == 'V1'
            hlr(i).LOC = 'V2';
            exdata = cellstr(hlr(i).LOC);
            where1 = 'where MSISDN = ''';
            where2 = msisdnstr;
            where3 = '''';
            whereclause = strcat(where1,where2,where3);
            update(conn, 'HLR', {'LOC'}, exdata, whereclause);
            break;
        else
            hlr(i).LOC = 'V1';
            exdata = cellstr(hlr(i).LOC);
            where1 = 'where MSISDN = ''';
            where2 = msisdnstr;
```

```
            where3 = "";
            whereclause = strcat(where1,where2,where3);
            update(conn, 'HLR', {'LOC'}, exdata, whereclause);
            break;
         end
      end
   end
result = 't';
return;
```

EVAG.M

```
function EvAg(n)
global time1
global time2
global time3
switch n
   case 1
      if time1 < fix(cputime)
         msc(2,n);
         time1 = time1 + 5;
         return;
      else
         return;
      end
   case 2
      if time2 < fix(cputime)
         msc(2,n);
         time2 = time2 + 5;
         return;
      else
         return;
      end
   case 3
      if time3 < fix(cputime)
```

```
            msc(2,n);
            time3 = time3 + 5;
            return;
        else
            return;
        end
end
```

REGISTERS_DATABASE.M

```
%Connect to database
logintimeout(5);
setdbprefs( 'DataReturnFormat','structure');
global conn
global curs1
global curs2
global curs3
global curs4
conn = database('Project', '', '');
%Retreive HLR table
curs1 = exec(conn, 'select * from HLR');
curs1 = fetch(curs1);
HLR = curs1.Data;
%Retreive VLR table
curs2 = exec(conn, 'select * from VLR');
curs2 = fetch(curs2);
VLR = curs2.Data;
%Retreive AUC table
curs3 = exec(conn, 'select * from AUC');
curs3 = fetch(curs3);
AUC = curs3.Data;
%Retreive EIR table
curs4 = exec(conn, 'select * from EIR');
curs4 = fetch(curs4);
EIR = curs4.Data;
```

```
%Copy onto global variables
global hlr
global vlr
global auc
global eir
hlr = cell2struct( [HLR.ID, HLR.IMSI, HLR.MSISDN, HLR.LOC, HLR.STATUS],
{'ID', 'IMSI', 'MSISDN', 'LOC', 'STATUS'}, 2 );
vlr = cell2struct( [VLR.ID, VLR.IMSI, VLR.MSISDN, VLR.HLR], {'ID', 'IMSI',
'MSISDN', 'HLR'}, 2 );
auc = cell2struct( [AUC.ID, AUC.IMSI, AUC.KP, AUC.ALD], {'ID', 'IMSI', 'KP',
'ALD'}, 2);
eir = cell2struct( [EIR.ID, EIR.IMEI, EIR.STATUS], {'ID', 'IMEI', 'STATUS'}, 2);
return;
```

VALIDATE.M

```
function status = validate()
status = 'not validated';
correctpin = {'1234'};
n = 1;
h = 0;
for i=1:3
   pinstr = inputdlg('Enter PIN(enter numbers): ');
   if cellfun('length',pinstr) ~= 4
      h(n) = warndlg('PIN should be 4 digits long');
      n = n+1;
   elseif strcmp(correctpin, pinstr)
      if h~= 0
         for m = 1:n
            close(h(m));
         end
      end
      status = 'validated';
      return;
   else
```

```
      h(n) = warndlg('Invalid PIN');
      n = n+1;
   end
end
for m = 1:(n-1)
   close(h(m));
end
n = 1;
h(n) = warndlg('PIN blocked');
n = n +1;
correctpuk = {'5678'};
for i = 1:10
   pukstr = inputdlg('Enter PUK to unblock: ');
   if cellfun('length',pukstr) ~= 4
      h(n) = warndlg('PUK should be 4 digits long');
      n = n+1;
   elseif strcmp(correctpuk, pukstr)
       for m = 1:(n-1)
          close(h(m));
       end
       status = 'validated';
       return;
   else
      h(n) = warndlg('Invalid PUK');
      n = n+1;
   end
end
 for m = 1:(n-1)
   close(h(m));
end
warndlg('PUK blocked');
status = 'not validated';
return;
```

ACC_DB.M

```
%Connect to database
logintimeout(5);
setdbprefs( 'DataReturnFormat','structure');
global conn
global curs5
% global fid
% conn = database('Project', '', '');
%Retreive ACCOUNT table
curs5 = exec(conn, 'select * from ACCOUNT');
curs5 = fetch(curs5);
ACC = curs5.Data;
%Copy onto global variables
global acc
acc = cell2struct( [ACC.lid, ACC.accno, ACC.bal], {'lid', 'accno', 'bal'}, 2 );
% fprintf(fid, '%s\n', 'Connected to db');
return;
```

LOGIN.M

```
function varargout = login(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @login_OpeningFcn, ...
                   'gui_OutputFcn',  @login_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
```

```
end


function login_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
set(hObject,'Name',num2str(varargin{1}));

function varargout = login_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function loginid_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function loginid_Callback(hObject, eventdata, handles)

function pwd_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function pwd_Callback(hObject, eventdata, handles)

function ok_Callback(hObject, eventdata, handles)
lid = get(handles.loginid, 'String');
pwd1 = get(handles.pwd, 'String');
switch lid
   case { 's123', 's456', 's789' }
```

```
    if strcmp(pwd1, lid) == 0
       warndlg( 'Invalid password' );
       set(handles.loginid, 'String', '');
       set(handles.pwd, 'String', '');
    else
       gcf = get(0,'CurrentFigure');
       mu = get(gcf,'Name');
       close(get(0,'CurrentFigure'));
       fundstransfer_appln(lid, mu);
    end
  otherwise
     warndlg( 'Invalid LoginId and password' );
     set(handles.loginid, 'String', '');
     set(handles.pwd, 'String', '');
end

function cancel_Callback(hObject, eventdata, handles)
global status1
global status2
global status3
global handles_copy
gcf = get(0,'CurrentFigure');
mu = str2num(get(gcf,'Name'));
switch mu
   case 1
      status1 = 'application closed';
   case 2
      status2 = 'application closed';
   case 3
      status3 = 'application closed';
end
if get(handles_copy.Mobile1,'Value')
   display_status(handles_copy, status1);
elseif get(handles_copy.Mobile2,'Value')
```

```
  display_status(handles_copy, status2);
elseif get(handles_copy.Mobile3,'Value')
  display_status(handles_copy, status3);
end
close(get(0,'CurrentFigure'));
```

FUNDSTRANSFER_APPLN.M

```
function varargout = login(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @login_OpeningFcn, ...
                   'gui_OutputFcn',  @login_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

function login_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
set(hObject,'Name',num2str(varargin{1}));

function varargout = login_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function loginid_CreateFcn(hObject, eventdata, handles)
if ispc
```

```
  set(hObject,'BackgroundColor','white');
else
  set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function loginid_Callback(hObject, eventdata, handles)

function pwd_CreateFcn(hObject, eventdata, handles)
if ispc
  set(hObject,'BackgroundColor','white');
else
  set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function pwd_Callback(hObject, eventdata, handles)

function ok_Callback(hObject, eventdata, handles)
lid = get(handles.loginid, 'String');
pwd1 = get(handles.pwd, 'String');
switch lid
  case { 's123', 's456', 's789' }
    if strcmp(pwd1, lid) == 0
      warndlg( 'Invalid password' );
      set(handles.loginid, 'String', '');
      set(handles.pwd, 'String', '');
    else
      gcf = get(0,'CurrentFigure');
      mu = get(gcf,'Name');
      close(get(0,'CurrentFigure'));
      fundstransfer_appln(lid, mu);
    end
  otherwise
    warndlg( 'Invalid LoginId and password' );
    set(handles.loginid, 'String', '');
```

```
    set(handles.pwd, 'String', '');
end

function cancel_Callback(hObject, eventdata, handles)
global status1
global status2
global status3
global handles_copy
gcf = get(0,'CurrentFigure');
mu = str2num(get(gcf,'Name'));
switch mu
  case 1
    status1 = 'application closed';
  case 2
    status2 = 'application closed';
  case 3
    status3 = 'application closed';
end
if get(handles_copy.Mobile1,'Value')
  display_status(handles_copy, status1);
elseif get(handles_copy.Mobile2,'Value')
  display_status(handles_copy, status2);
elseif get(handles_copy.Mobile3,'Value')
  display_status(handles_copy, status3);
end
close(get(0,'CurrentFigure'));
```

FLIGHTTICKET_RESERVATION.M

```
function varargout = flightticket_appln(varargin)
gui_Singleton = 0;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @flightticket_appln_OpeningFcn, ...
                   'gui_OutputFcn',  @flightticket_appln_OutputFcn, ...
```

```
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

function flightticket_appln_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
set(handles.logdata, 'String', 'Application Started');
set(hObject,'Name',num2str(varargin{1}));
guidata(hObject, handles);

function varargout = flightticket_appln_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function fname_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function fname_Callback(hObject, eventdata, handles)

function date_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
```

```
end


function date_Callback(hObject, eventdata, handles)

function class_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function class_Callback(hObject, eventdata, handles)

function source_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function source_Callback(hObject, eventdata, handles)

function dest_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function dest_Callback(hObject, eventdata, handles)

function ok_Callback(hObject, eventdata, handles)
fname1 = 'Air Deccan';
```

```
fname2 = 'Air India';
fname3 = 'Indian Airlines';
fname4 = 'Indian';
fname5 = 'Air Sahara';
class1 = 'First class';
class2 = 'Business class';
class3 = 'Economic class';
source1 = 'DELHI';
source2 = 'MUMBAI';
source3 = 'KOLKATTA';
source4 = 'CHENNAI';
source5 = 'BANGALORE';
dest1 = 'DELHI';
dest2 = 'MUMBAI';
dest3 = 'KOLKATTA';
dest4 = 'CHENNAI';
dest5 = 'BANGALORE';
flight = fopen( 'flight.txt', 'a' );
value = get(handles.fname,'Value');
switch value
   case 1
      fname = fname1;
   case 2
      fname = fname2;
   case 3
      fname = fname3;
   case 4
      fname = fname4;
   case 5
      fname = fname5;
end
value = get(handles.class,'Value');
switch value
   case 1
```

```
        class = class1;
    case 2
        class = class2;
    case 3
        class = class3;
end
date1 = get(handles.date, 'String');
seats = get(handles.seats, 'String');
value = get(handles.source,'Value');
switch value
    case 1
        source = source1;
    case 2
        source = source2;
    case 3
        source = source3;
    case 4
        source = source4;
    case 5
        source = source5;
end
value = get(handles.dest,'Value');
switch value
    case 1
        dest = dest1;
    case 2
        dest = dest2;
    case 3
        dest = dest3;
    case 4
        dest = dest4;
    case 5
        dest = dest5;
end
```

```
fprintf(flight, '%s\n',fname);
fprintf(flight, '%s\n',date1);
fprintf(flight, '%s\n',class);
fprintf(flight, '%s\n',seats);
fprintf(flight, '%s\n',source);
fprintf(flight, '%s\n',dest);
fprintf(flight, '\n');
fclose(flight);
logdata = get(handles.logdata, 'String');
logdata = strvcat( logdata, 'Ticket Reserved');
logdata = strvcat( logdata, 'Flight: ', fname);
logdata = strvcat( logdata, 'Date: ', date1);
logdata = strvcat( logdata, 'Class: ', class);
logdata = strvcat( logdata, 'No of tickets reserved: ', seats);
logdata = strvcat( logdata, 'Source: ', source);
logdata = strvcat( logdata, 'Destination: ', dest);
set(handles.logdata, 'String', logdata);

function cancel_Callback(hObject, eventdata, handles)
global time1
global time2
global time3
global msisdnstr1
global msisdnstr2
global msisdnstr3
global status1
global status2
global status3
global handles_copy
gcf = get(0,'CurrentFigure');
mu = str2num(get(gcf,'Name'));
switch mu
   case 1
      muno = msisdnstr1;
```

```
        status1 = 'application closed';
        EvAg(1);
        time1 = time1 + 3;
    case 2
        muno = msisdnstr2;
        status2 = 'application closed';
        EvAg(2);
        time2 = time2 + 3;
    case 3
        muno = msisdnstr3;
        status3 = 'application closed';
        EvAg(3);
        time3 = time3 + 3;
end
if get(handles_copy.Mobile1,'Value')
    display_status(handles_copy, status1);
elseif get(handles_copy.Mobile2,'Value')
    display_status(handles_copy, status2);
elseif get(handles_copy.Mobile3,'Value')
    display_status(handles_copy, status3);
end
logdata = get(handles.logdata, 'String');
logdata = strvcat( logdata, 'Application closed');
set(handles.logdata, 'String', logdata);
str = strcat(muno,'.txt')
log1 = fopen( str, 'a' );
fprintf(log1, '%s\n', 'Flight_Ticket_Reservation_Application');
fprintf(log1, '%s\n',date);
c = fix(clock);
time = strcat(num2str(c(4)), ':', num2str(c(5)), ':', num2str(c(6)));
fprintf(log1, '%s\n',time);
[m,n] = size(logdata);
for i = 1:m
    fprintf(log1, '%s\n',logdata(i,:));
```

```
end
fprintf(log1,'\n');
fclose(log1);
close(get(0,'CurrentFigure'));

function seats_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function seats_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function logdata_CreateFcn(hObject, eventdata, handles)
% hObject    handle to logdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in logdata.
function logdata_Callback(hObject, eventdata, handles)
% hObject    handle to logdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns logdata contents as cell array
%        contents{get(hObject,'Value')} returns selected item from logdata
```

WEATHERREPORT_APPLN.M

```
function varargout = weatherreport_appln(varargin)
gui_Singleton = 0;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @weatherreport_appln_OpeningFcn, ...
                   'gui_OutputFcn',  @weatherreport_appln_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

function weatherreport_appln_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
set(hObject,'Name',num2str(varargin{1}));
set(handles.maxtemp, 'String', '25C');
set(handles.mintemp, 'String', '20C');
set(handles.forecast, 'String', 'Cloudy');
set(handles.logdata, 'String', 'Application Started');
guidata(hObject, handles);

function varargout = weatherreport_appln_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function places_CreateFcn(hObject, eventdata, handles)
if ispc
```

```
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function places_Callback(hObject, eventdata, handles)
place1 = 'DELHI';
place2 = 'MUMBAI';
place3 = 'CHENNAI';
place4 = 'BANGALORE';
place5 = 'KOLKATTA';
place = 'DELHI';
value = get(handles.places,'Value');
switch value
   case 1
      max = '25C';
      min = '20C';
      forecast = 'Cloudy';
      place = 'DELHI';
   case 2
      max = '30C';
      min = '28C';
      forecast = 'Sunny';
      place = 'MUMBAI';
   case 3
      max = '30C';
      min = '29C';
      forecast = 'Rain';
      place = 'CHENNAI';
   case 4
      max = '28C';
      min = '26C';
      forecast = 'Heavy Rain with thunder';
      place = 'BANGALORE';
```

```
   case 5
      max = '32C';
      min = '29C';
      forecast = 'Sunny';
      place = 'KOLKATTA';
end
set(handles.maxtemp, 'String', max);
set(handles.mintemp, 'String', min);
set(handles.forecast, 'String', forecast);
logdata = get(handles.logdata, 'String');
logdata = strvcat( logdata, 'Viewed Weather report for place: ', place);
set(handles.logdata, 'String', logdata);

function close_Callback(hObject, eventdata, handles)
global time1
global time2
global time3
global msisdnstr1
global msisdnstr2
global msisdnstr3
global status1
global status2
global status3
global handles_copy
gcf = get(0,'CurrentFigure');
mu = str2num(get(gcf,'Name'));
switch mu
   case 1
      muno = msisdnstr1;
      status1 = 'application closed';
      EvAg(1);
      time1 = time1 + 3;
   case 2
      muno = msisdnstr2;
```

```
        status2 = 'application closed';
        EvAg(2);
        time2 = time2 + 3;
    case 3
        muno = msisdnstr3;
        status3 = 'application closed';
        EvAg(3);
        time3 = time3 + 3;
end
if get(handles_copy.Mobile1,'Value')
    display_status(handles_copy, status1);
elseif get(handles_copy.Mobile2,'Value')
    display_status(handles_copy, status2);
elseif get(handles_copy.Mobile3,'Value')
    display_status(handles_copy, status3);
end
logdata = get(handles.logdata, 'String');
logdata = strvcat( logdata, 'Application closed');
set(handles.logdata, 'String', logdata);
str = strcat(muno,'.txt')
log1 = fopen( str, 'a' );
fprintf(log1, '%s\n', 'Weather_Report_Application');
fprintf(log1, '%s\n',date);
c = fix(clock);
time = strcat(num2str(c(4)), ':', num2str(c(5)), ':', num2str(c(6)));
fprintf(log1, '%s\n',time);
[m,n] = size(logdata);
for i = 1:m
    fprintf(log1, '%s\n',logdata(i,:));
end
fprintf(log1,'\n');
fclose(log1);
close(get(0,'CurrentFigure'));
```

```
% --- Executes during object creation, after setting all properties.
function logdata_CreateFcn(hObject, eventdata, handles)
% hObject    handle to logdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in logdata.
function logdata_Callback(hObject, eventdata, handles)
% hObject    handle to logdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns logdata contents as cell array
%        contents{get(hObject,'Value')} returns selected item from logdata
```

ABOUT.M

```
function varargout = about(varargin)
% ABOUT M-file for about.fig
%      ABOUT, by itself, creates a new ABOUT or raises the existing
%      singleton*.
%
%      H = ABOUT returns the handle to a new ABOUT or the handle to
%      the existing singleton*.
%
%      ABOUT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ABOUT.M with the given input arguments.
%
%      ABOUT('Property','Value',...) creates a new ABOUT or raises the
```

```
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before about_OpeningFunction gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to about_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help about
% Last Modified by GUIDE v2.5 25-Mar-2002 12:19:15
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @about_OpeningFcn, ...
                   'gui_OutputFcn',  @about_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before about is made visible.
function about_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to about (see VARARGIN)

% Choose default command line output for about
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes about wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = about_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
```

HELP1.M

```
function varargout = help1(varargin)
% HELP1 M-file for help1.fig
%      HELP1, by itself, creates a new HELP1 or raises the existing
%      singleton*.
%
%      H = HELP1 returns the handle to a new HELP1 or the handle to
%      the existing singleton*.
%
%      HELP1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in HELP1.M with the given input arguments.
%
%      HELP1('Property','Value',...) creates a new HELP1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before help1_OpeningFunction gets called.  An
```

```
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to help1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help1 help1
% Last Modified by GUIDE v2.5 15-Mar-2006 17:42:51
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @help1_OpeningFcn, ...
                   'gui_OutputFcn',  @help1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before help1 is made visible.
function help1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to help1 (see VARARGIN)
```

```
% Choose default command line output for help1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes help1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = help1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
```

8.2 SAMPLE OUTPUT

Figure 8.1 Splash screen

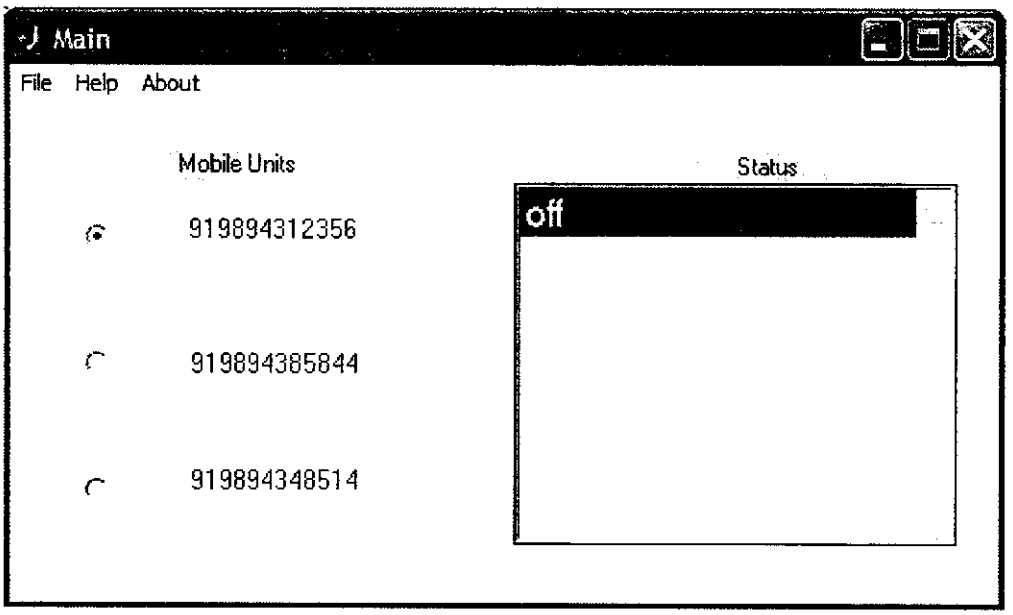

Figure 8.2 Main screen

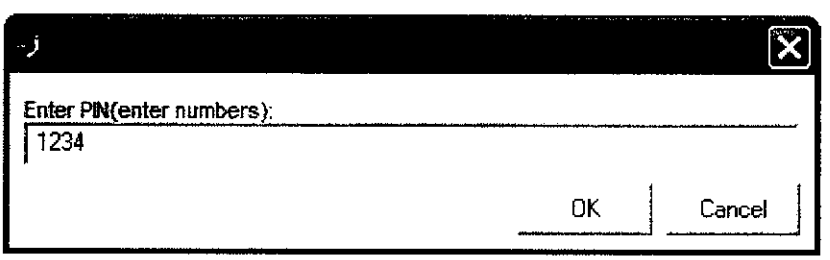

Figure 8.3 PIN Request

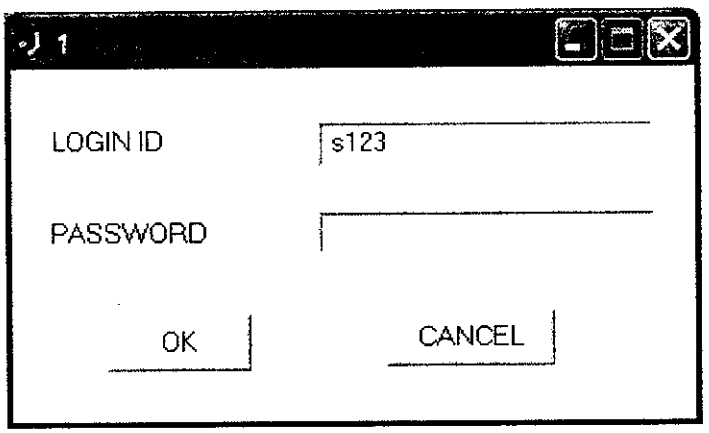

Figure 8.4 Login screen

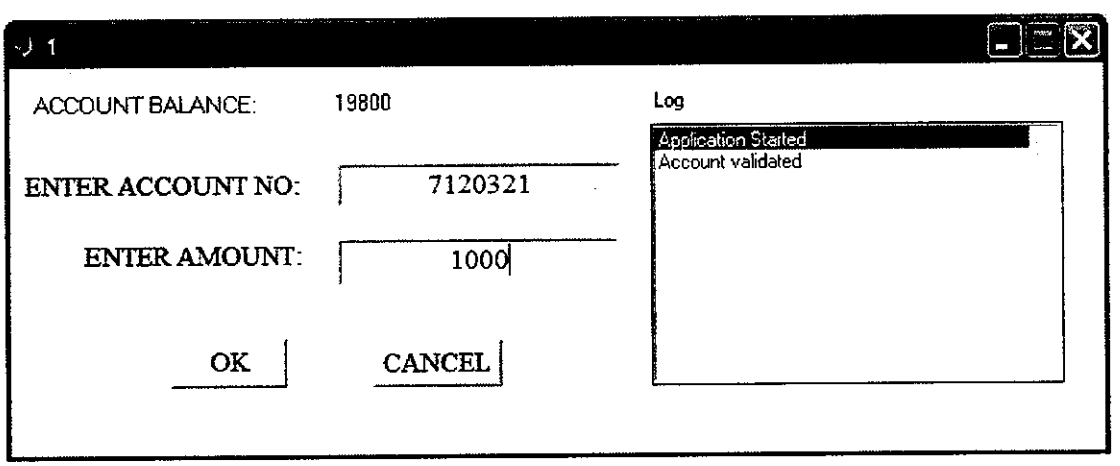

Figure 8.5 Funds Transfer screen

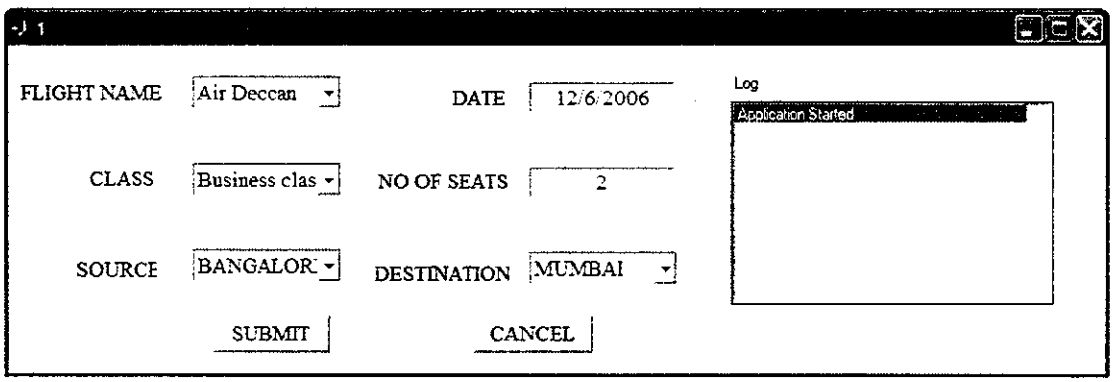

Figure 8.6 Flight Ticket Reservation screen

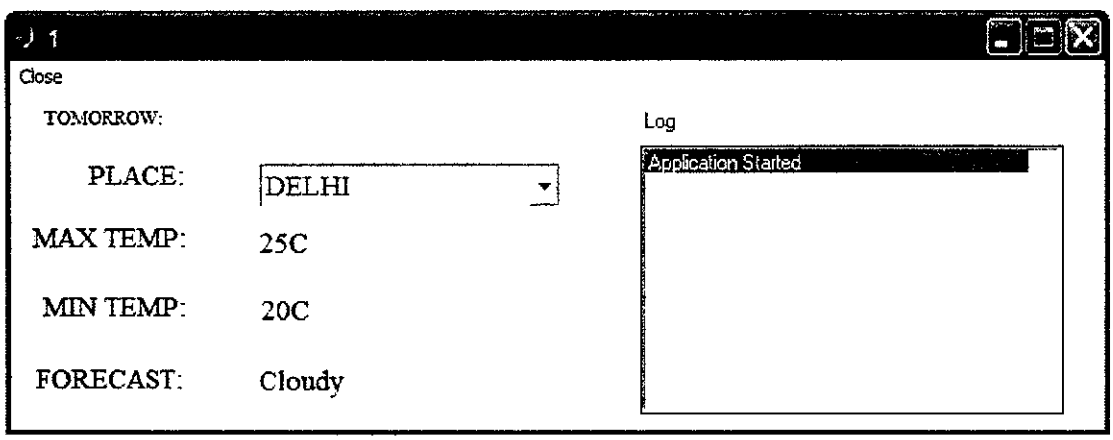

Figure 8.7 Weather Report screen

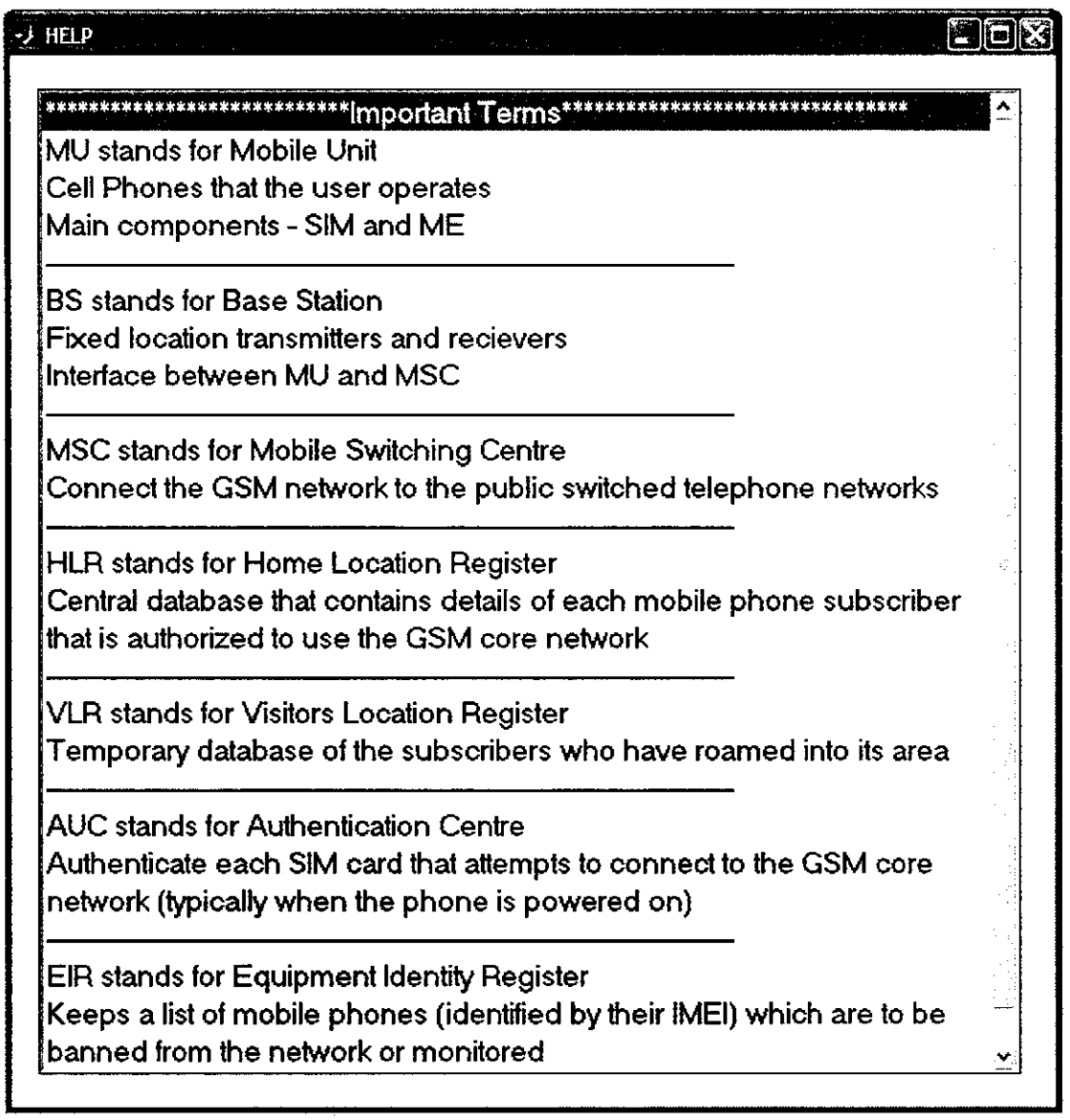

Figure 8.8 Help screen

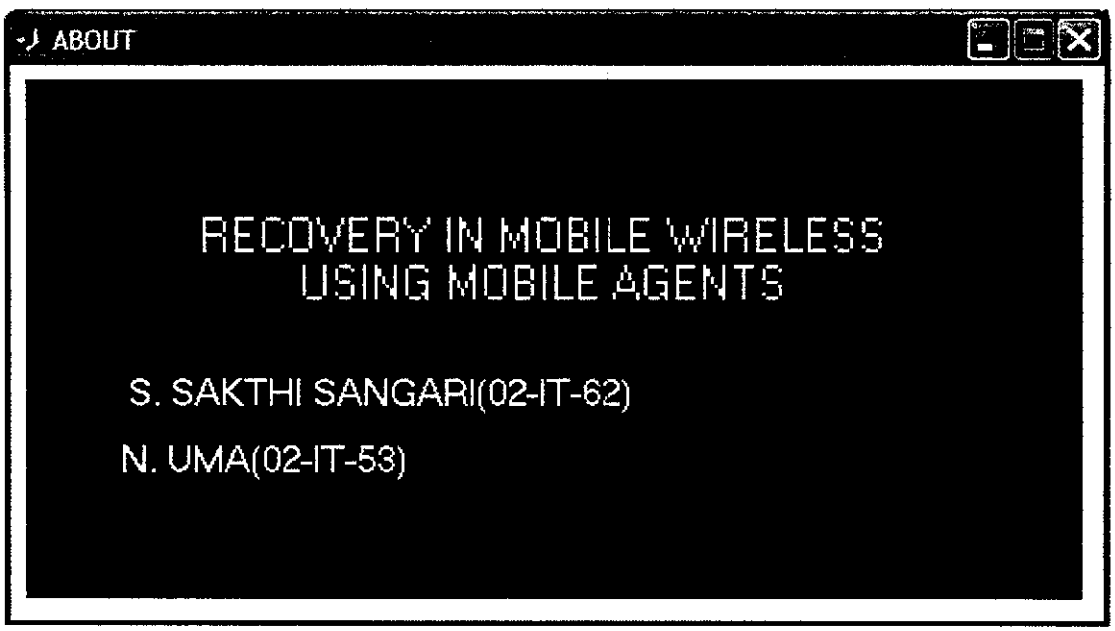

Figure 8.9 About screen

9. REFERENCES

1. C. Dhawan, "Mobile Computing", McGraw, pp.13-68, 1997.

2. S. Gadiraju and V. Kumar, "Recovery in the Mobile Wireless Environment Using Mobile Agents," IEEE Transactions On Mobile Computing, vol. 3, no. 2, 2004.

3. D. Kotz, R. Gray, S. Nog, D. Rus, S. Chawla, and G. Cybenko, "AgentTCL: Targeting the Needs of Mobile Computers," IEEE Internet Computing, vol. 1, no. 4, 1997.

4. C. Perkins, "Mobile Networking through Mobile IP," IEEE Internet Computing, pp. 58-69, Jan. 1998.

5. R. Pratap, "Getting Started with MATLAB", Oxford University Press, 1998.

6. R. Steele, Chin_Chun Lee, P. Gould, "GSM, cdmaOne and 3G Systems", Wiley, 2001