CHAPTER 6 IMPLEMENTATION AND APPLICATION

6.1 INTRODUCTION

In this chapter we discuss about the implementation of proposed solution and the entire system is realized from a broader concept to the actual working system. It shows how the objectives and features of the system are actually accomplished with the working system. In the previous chapter an overview of the vehicle tracking system architecture was introduced and has identified the main system components as objects with associated interfaces. In general, an object oriented view of software systems through all modeling and design phases permits a smooth transition from high-level system description to increasingly in-depth and complex architecture models. Therefore, the subsequent discussion of both the vehicle tracking system and the location based service architecture benefit from the component oriented discussion of the system architecture in the previous sections and represents a refined software model that allows for immediate realization in an object-oriented implementation language.

6.2 IMPLEMENTATION LANGUAGE

The language chosen for the implementation of this tracking software is Java ™, developed by Sun Microsystems Inc. It features a clear language structure and is deployable on many different computing platforms through a software layer called the Java Virtual Machine (JVM). The JVM interprets Java ™ applications and transforms them into platform specific executable code. Advantages of selecting such conventional fourth generation language are memory management, error handling and multiple threading. And finally, it is really important when considering the scalability requirements of the vehicle tracking system as well as the necessity to share data and communication resources in an asynchronous manner.

6.3 IMPLEMENTATION ENVIRONMENT

In the analysis phase it was identified required hardware and software environments that the proposed application will be implemented. During the prototype and the software implementation phases, it was used a server class PC with even higher capacity of RAM (1 GB) to attain better performance of the system since this tracking application is comparatively process intensive. Windows XP professional with service

pack-2 were installed on the machine. Then it was installed Java standard edition 5.0 Development Kit (*JDK 5.0*) and required libraries for java tracking application. It is shown all required java libraries for implementation of this tracking application in the *Appendix A*. Then it was installed and configured the database *MySQL-essential-5.0.15-win32*. Then *Eclipse-SDK-3.1.1* was installed and used as Java (IDE) Integrated Development Environment for the implementation of tracking application.

The monitoring station machine was connected to a broad band internet connection as it needs to have the communication with LBS to collect required real-time positioning information of tracking vehicles. Since the monitoring station machine was exposed to the Internet, it was brought behind to a *Checkpoint* firewall system to prevent from security vulnerabilities and harden up the safety measures.

6.4 THE TRACKING APPLICATION

The particular vehicle tracking application was purely built up on the above mentioned system environment. The classes that were identified in the design phase should be coded and implemented in this implementation phase.

6.4.1 USER MANIPULATION AND AUTHORIZATION

The system provides a login window to tracking user at the initial stage, when the user interacts with the system. *LoginWindow* window is used to collect login information from user. Only authorized users are allowed to use the system. Authorized users can be defined utilizing the *UserWindow*. This window is also used to create new users who are allowed to use the tracking system. The *User* is used to manipulate and handling of user logins to the system.

DBConnection is one of the important classes in the application as it manages database connections, handle of SQL queries etc. The following method is used to get the database connection.

"private static synchronized Connection getConnection() throws Throwable {

```
Connection connection = null;
Class.forName(DB_DRIVER);
connection = DriverManager.getConnection(DB_URL, DB_LOGIN,
DB_PASSWORD);
```

```
        return connection;
    }"
```

6.4.2 MAIN WINDOW IMPLEMENTATION

The *MainWindow* is the main window of the system and acts as the container for all other windows whereas *MainWorkspace* window is used to track vehicles and this window acts as the container for *ControlPanel* and the *Map* windows.

"MainWorkspace.doClickSearchButton ()" method is used to retrieve vehicle coordinates. This sends an http request to the web service implemented in LBS system. Then the location information is wrapped as an xml and will be sent back to the tracker. Following segment of code illustrates the method which is used.

```
"   Private void doClickSearchButton () {
        //tracker get table model
        TrackerTableModel model = (TrackerTableModel)
m_controlPanel.m_trackingTable.getModel();

        String mainAlert = "";

        for (int i = 0; i < model.getRowCount(); i++) {
            String vehicleNo = (String) model.getValueAt(i,0);
            String trackerNo = (String) model.getValueAt(i,1);
            String color = (String) model.getValueAt(i,2);


        String xml = "";

        progressMonitor.setProgress(10);

        String mobileNo = trackerNo; //m_controlPanel.getMobileNo();
        String trackerURL = TrackerToolkit.TRACKER_SERVICE_URL
                + "username=" +
TrackerToolkit.TRACKER_SERVICE_UID
                + "&password="
                + TrackerToolkit.TRACKER_SERVICE_PASSWORD
                + "&clientcode="
                +
TrackerToolkit.TRACKER_SERVICE_CLIENT_CODE
                + "&phoneno=" + mobileNo;

        System.out.println(trackerURL);
        Location location = null;

        if (xml.length() <= 0) {
```

```
} else {
        location = TrackerToolkit.getLocation(xml);
}"
```

The *VehicleWindow* is used to add and remove new vehicles, which need to be tracked, where as *Track* is used to manipulate vehicle tracking records.

6.4.3 BASE MAP IMPLEMENTATION

The java class "*Map*" is used to render shape files and to show tracking locations on the base map.

Method called "initializeMap ()" is used to render different map layers that are stored in shape files and create styles like colors, line width and shapes for different layers. Method "addCoordinates (ArralyList)" is used to plot locations of tracking vehicles on the base map, while "isInsideRestrictedArea" method returns "true" if a given location of a tracking vehicle is within a particular restricted area. Method "addPolygon (RestrictedArea)" is used to draws restricted areas (polygons) on the base map.

ControlPanel window contains information such as the list of current vehicles that are being tracked, restricted polygon information and also holds the search and other control buttons.

The *Tracker Toolkit* class is used to parse an XML received from the LBS and to extract location information about tracking vehicles. The *Location* class represents the tracking coordinates.

Both *Restricted Area* and *RestrictedAreaWindow* classes are used to manipulate restricted areas and add and remove restricted areas respectively.

6.4.4 CONTROL PANEL FEATURES

There are several supporting classes used in order to implement features exhibit on the control panel of the tracking application. The *HistoryTableModel* is used as table model for the history table in control panel. And *RestrictedAreaTableModel* is used as table model for the restricted area table in control panel. To implement tracker table model in control panel *TrackerTableModel* is utilized while *TrackerTableRenderer* operates as common table renderer for all tables used in control panel.

6.5 TRACKING GUI WINDOW

Following is the final outcome of the vehicle tracking application main GUI window. It contains all the functions and features that are attentively distributed on the main GUI control panel. The entire functionality of the vehicle tracking system is depicted with relevant screen layouts in the *Appendix D*.

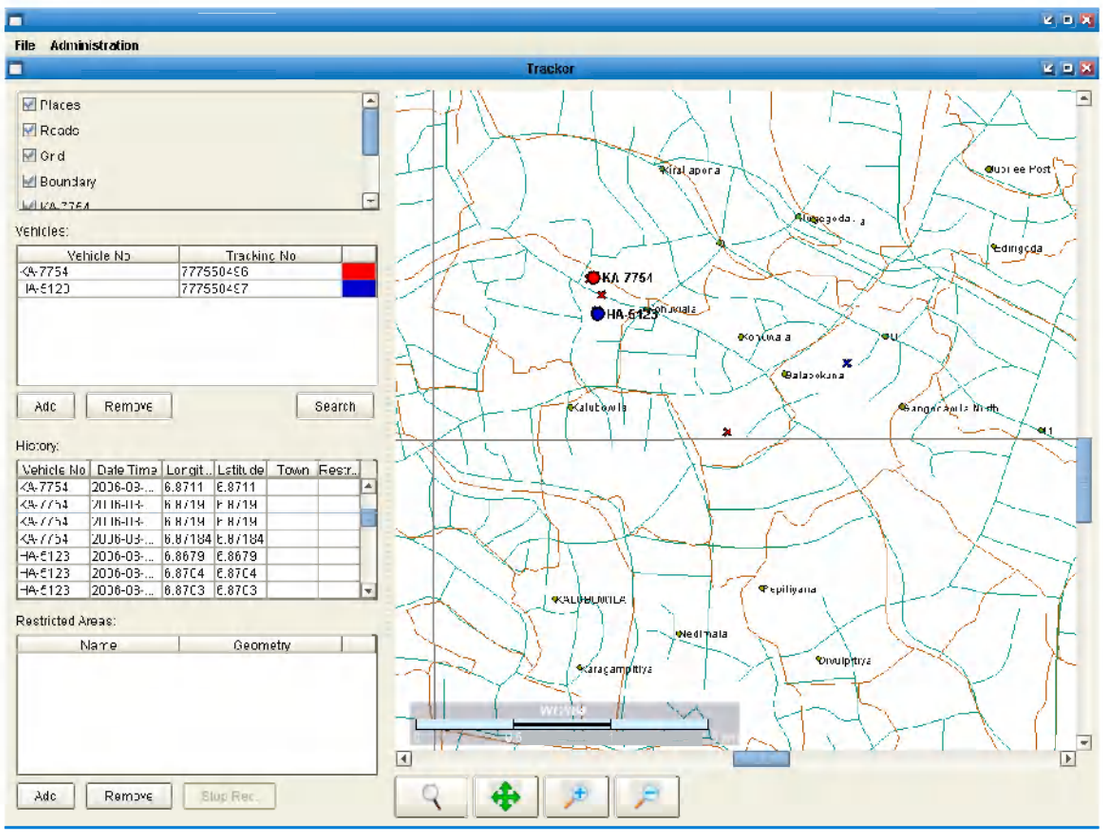

Figure 6-1: Tracking GUI window

6.6 IMPLEMENTATION OF CAR UNIT

The car unit is purely a hardware component which is industry developed and freely available in the market at an affordable price. In this particular pilot project implementation, it is used a *Wavecom*[3] *FASTRACK* dual band GSM/GPRS modem (M1306B) with a phase-II Tracking enable SIMs. That modem was designed for data, fax, SMS and voice applications. The output power of the device is 2W at 900MHz and 1W at 1800MHz respectively. Input voltage between 5V to 32V. The device is housed in rugged casing and it is robust for a hard use. In the pilot project this device is properly fixed on the tracking vehicle. Following figure 6-2 shows the *WaveCom FastTrack* GSM/GPRS modem which is considered to use for the car unit.

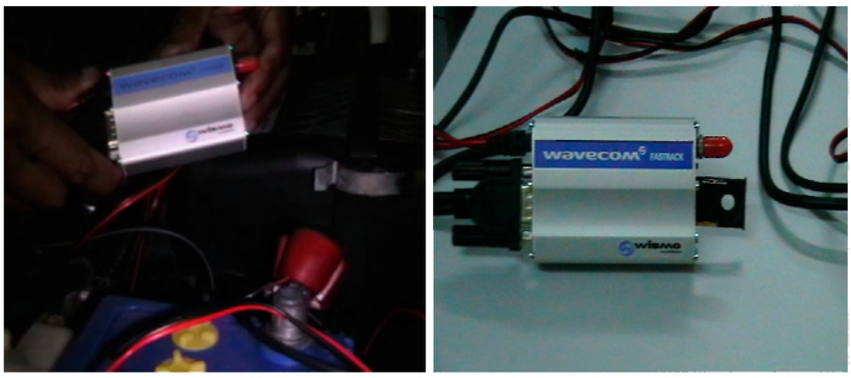

Figure 6-2: WaveCom FastTrack GSM/GPRS modem

6.7 SUMMARY

In this chapter was discussed about the implementation phase of the proposed vehicle tracking system in order to demonstrate how the entire application is systematize. We talked about the advantages of object-oriented language and Java as the application development language. Then it was discussed, the environment of hardware and software that the vehicle tracking application is deployed, and it was also elaborated communication and security aspects for the final deliverable. To make it easy for understanding and elaboration purposes, the implementation of vehicle tracking application was categorized into several sub topics and discussed each of them by providing required supplementary materials in the appendixes. Finally it was talked about hardware component which will be mounted on tracking vehicle as the car unit. In the next chapter, it is intended to talk about testing and evaluation of the vehicle tracking application.