



Design Document

IT CARLOW

Traffic Tracking Application on Android

Xufeng Li

C00131008

Table of Contents

1. Introduction	3
2. Data Structures Design	4
2.1 Database Table	4
2.2 Process KML Data File	6
3. Architectural Design.....	8
4. Algorithm Design.....	10
5. Module Design.....	14
5.1 Overview Dependencies between Modules	14
5.2 Brief Responsibilities of Major Module	15
6. Conclusions.....	25

1. Introduction

This document indicates the software design of Traffic Tracking Application (TTA). The TTA is an Android platform application, the aim of it is keep safe driving and avoid road fatalities. The TTA can track user's driving route and speed, it warns users when their speed out of limit. And the data of driving also can be transferred into a website for checking user's driving behaviors.

In this document, it includes data structures design, architectural design, module design and human interfaces design. The blueprint of TTA is shown from this document, coding can be followed with this design. The detail of project's feature is also discussed again in order to design of TTA. Here is a list of features:

✧ User's function features

- register account
- login/logout
- view map and speed
- speeding tracking
- route tracking
- view data of trips

✧ Manager's function features

- login/logout
- view data of trips
- manage users

2. Data Structures Design

The TTA chooses SQLITE as database to store data about app and uses Spatialite [1] to extension enables SQLITE to support spatial data. Such as POINT, LINESTRING, etc. The data structure is shown on database table. Here is a list of them. (More details in module design)

2.1 Database Table

Tbl_user		
Field name	Data Type	Comments
U_id	INTEGER	Primary key
U_user	NVARCHAR	User's login ID
U_code	NVARCHAR	User's password
U_mail	NVARCHAR	e-mail address
U_admin	INTEGER	1 is administrator,0 is common user
U_time	TIMESTAMP	Datetime of register

* The table of tbl_user is used to store account's information.

Tbl_road		
Field name	Data Type	Comments
R_id	INTEGER	Primary key
R_name	NVARCHAR	Name of this road
R_location	NVARCHAR	Name of location

* The table of tbl_road is used to keep data of road

Tbl_coords_road		
Field name	Data Type	Comments
C_id	INTEGER	Primary key
R_id	INTEGER	Primary key of road
C_coord_l	FLOAT	Longitude of location
C_coord_d	FLOAT	Dimension of location
C_speed_limt	INTEGER	Speed limit

* The table of tbl_coords_road is used to store location of speed trips.

Tbl_driving_route		
Field name	Data Type	Comments
D_id	INTEGER	Primary key
D_user	NVARCHAR	User's login ID
D_start	NVARCHAR	Start point of this route
D_end	NVARCHAR	End point of this route
D_ave_speed	INTEGER	Average of driving speed
D_max_speed	INTEGER	Max speed of driving
D_time	TIMESTAMP	Datetime of information be updated
D_via	LINestring	Location of via point

* The table of tbl_driving_route is used to keep data of user driving route.

2.2 Process KML Data File

The project has a KML file which stores location of speed camera tracking route in Ireland. The KML file is a file format which designed for Google Map and Google Earth. It uses XML syntax to display geographic information. User can create it to pinpoint locations, add image overlays, etc. In Traffic Tracking App, it chooses SAXReader [3] to parse KML and gets coordinates of speed zone. Here is a part of code which about parse KML by SAXReader in Java.

```
DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
Document doc = docBuilder.parse (new File("gps.kml"));

// normalize text representation
doc.getDocumentElement ().normalize ();
System.out.println ("Root element of the doc is " +
    doc.getDocumentElement ().getNodeName ());

NodeList listOfPersons = doc.getElementsByTagName ("Placemark");
int totalPersons = listOfPersons.getLength();
System.out.println("Total no of Placemark : " + totalPersons);

for(int s=0; s<listOfPersons.getLength()-500 ; s++){

    Node firstPersonNode = listOfPersons.item(s);
    if (firstPersonNode.getNodeType() == Node.ELEMENT_NODE){

        Element firstPersonElement = (Element)firstPersonNode;
        NodeList firstNameList = firstPersonElement.getElementsByTagName ("coordinates");
        Element firstNameElement = (Element)firstNameList.item(0);

        NodeList textFNList = firstNameElement.getChildNodes();
        System.out.println("The coordinates are : " +
            ((Node)textFNList.item(0)).getNodeValue().trim());
    }
}
```

(Parser KML in Java)

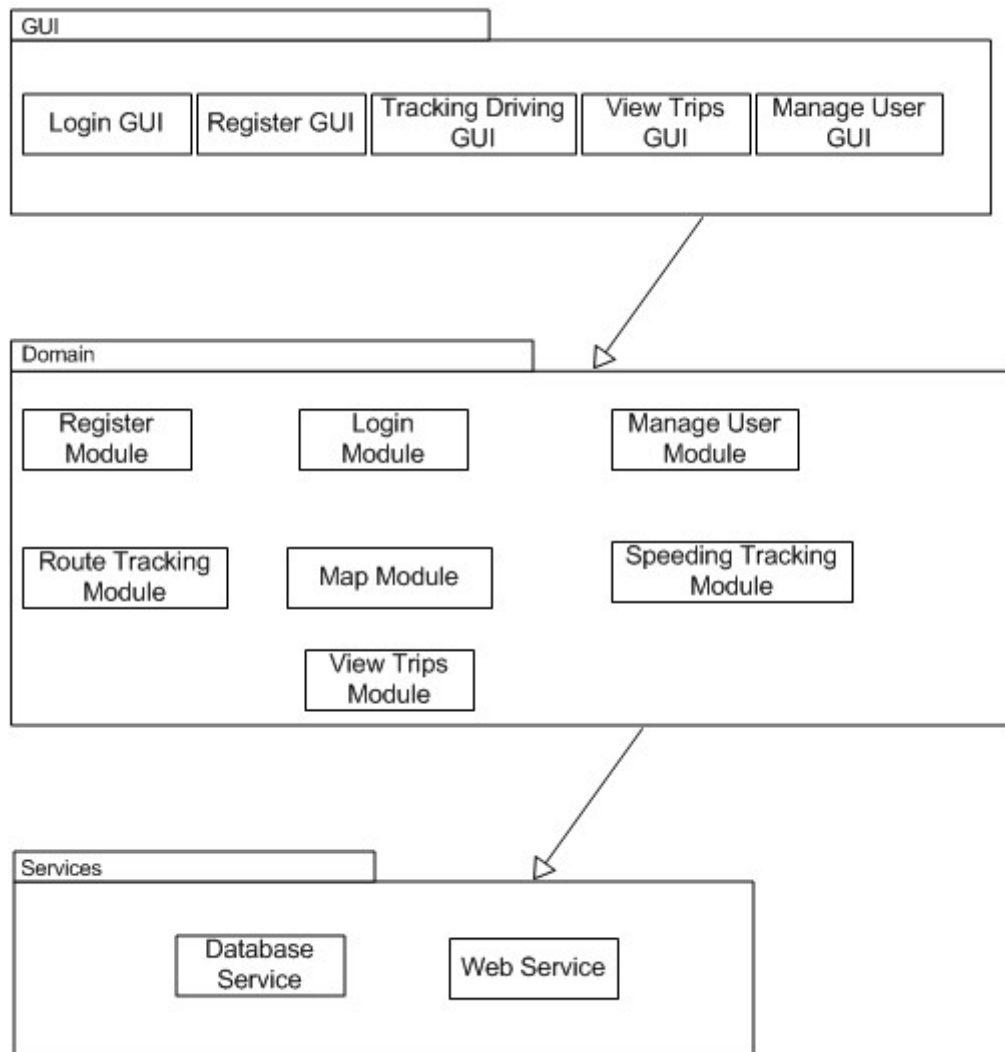
The code open the KML file and find element which named by 'coordinates'. Then results are displayed. The result of this code is described in next diagram:

```
Console Problems
<terminated> kml [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2011-1-24 下午10:11:3)
Root element of the doc is kml
Total no of Placemark : 515
The coordinates are :-9.71122630131129,52.2651490879693,0
                    -9.70832564920696,52.2690180702523,0
                    -9.70854305821369,52.2698653611445,0
                    -9.7073788876339,52.2720366447056,0
                    -9.70644166272694,52.2729006325629,0
                    -9.70274785291802,52.2753918845182,0
                    -9.70277986091443,52.2774324685784,0
                    -9.70238094073869,52.2790823967509,0
The coordinates are :-9.76751213573442,52.279988352698,0
                    -9.76951294011875,52.2791646341502,0
                    -9.77138296552796,52.278153856091,0
                    -9.77364201935106,52.2776285710961,0
                    -9.77852946288182,52.2765723066995,0
                    -9.7814415150229,52.2754322711193,0
                    -9.78556442811876,52.2752943885903,0
                    -9.7899458124756,52.2754548238404,0
                    -9.79737265773925,52.2747372204736,0
                    -9.80302726676018,52.2743864083192,0
                    -9.8096300796658,52.2746255563182,0
```

(Result of code)

The app gets coordinates from this Java class, and then updates these coordinates into database table separately. These data could be used by modules to tracking user's driving,

3. Architectural Design



(System Architecture)

The TTA system architecture is typical layered architecture. The system is grouped into three layers, GUI layers, Domain layers and Services layers. Communications of them in top-down fashion,

The TTA system GUI layer consists of five modules. The Login GUI, Register GUI and Tracking Driving GUI are designed for Android platform which used by drivers to operate the system on the cell phone. The View Trips GUI and Manage User GUI belong to web site which can be visited by users to check

driving data from computer's browsers.

The TTA system Domain layer contains seven modules. The Register module, Login module and Manage User module deal with user's account. The Route Tracking module, Map module, Speeding Tracking module and View Trips module are key modules in TTA. The route tracking and speeding racking are based on map module, the map module displays tracking information to users.

The Services layer has two services. The Database service is in charge of the whole data in TTA. The Web service displays information of trips to users and user's account to manager.

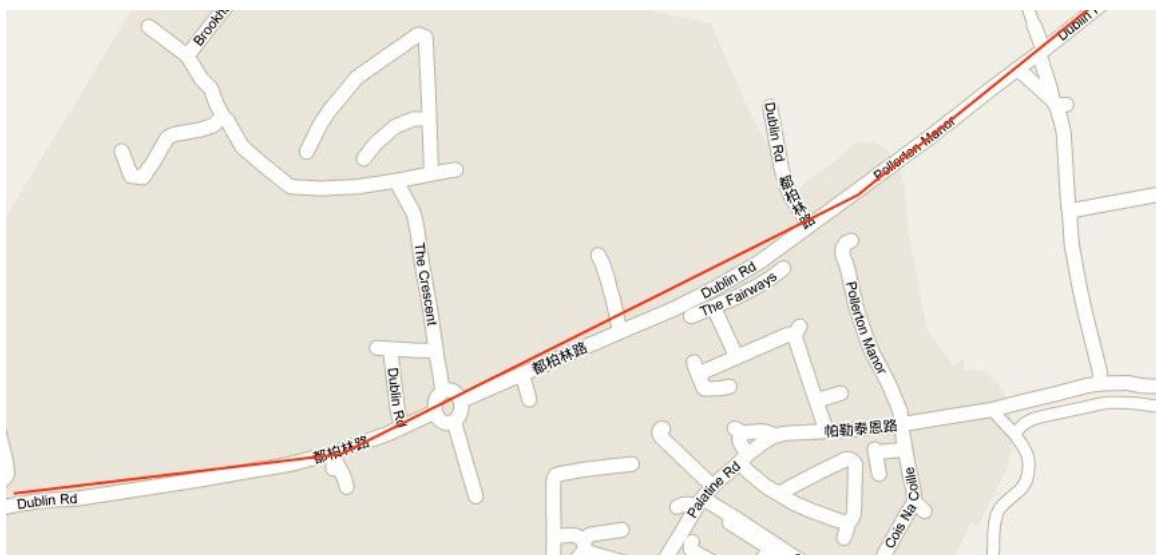
4. Algorithm Design

The algorithm of Tracking Traffic App is used for matching coordinate of driver with speed camera zone. The app keeps speed camera coordinates in database. These zones originate from garda's website [2]:



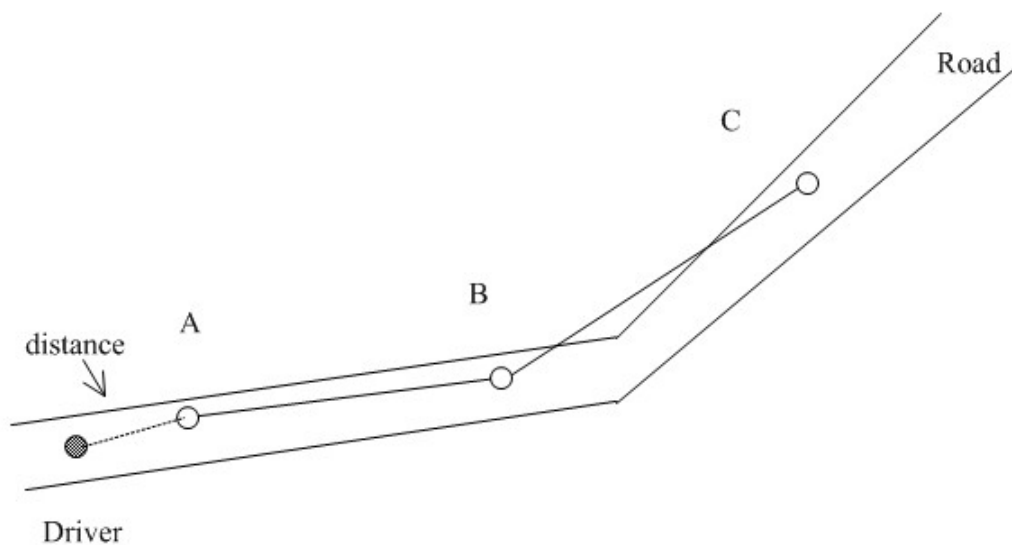
(Speed camera zone in Carlow)

The red lines on above picture are location of zone in Carlow. One of them enlarged in the second diagram.



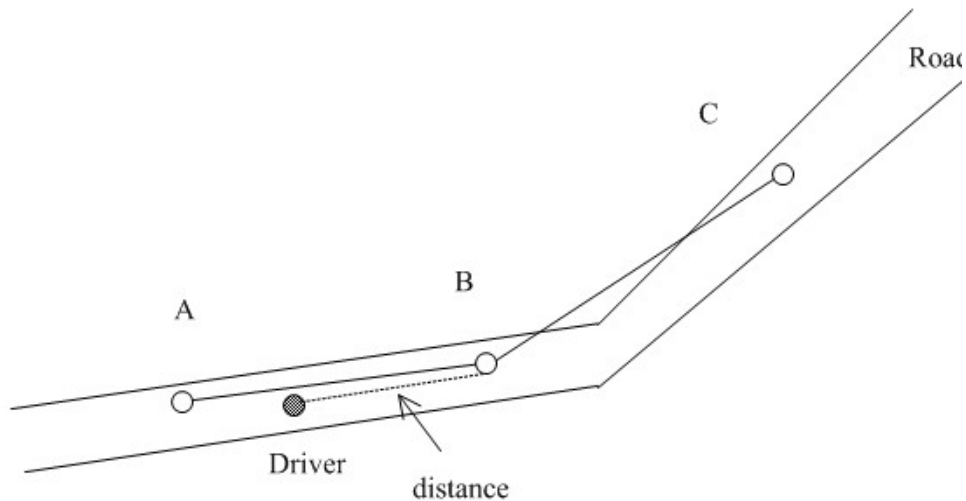
In this diagram, it clearly represents some part of speed tracking route does not coincide with road (such as Dublin Rd). Because these route are stored as some via point in database. So the app needs an algorithm to snap driver's location with these routes during speeding tracking. If app can estimate a point in the route or near the route, this algorithm is successful.

The algorithm's steps:



(algorithm_1)

1. The white circles (A, B, C) are via point of speed zone on the road. The black circle (Driver) is location of driver. When app gets driver's location from GPS, it selects data of speed zone in database, if there has a speed zone in the same road, app keeps location of zone and begin algorithm.
2. Firstly, algorithm calculates the distance between driver's location and start point A. If it less than 5 meters, it also means the driver is almost enter speed zone and the app begins to give a warning. This is a condition of driver enter speed zone. (see algorithm_1)

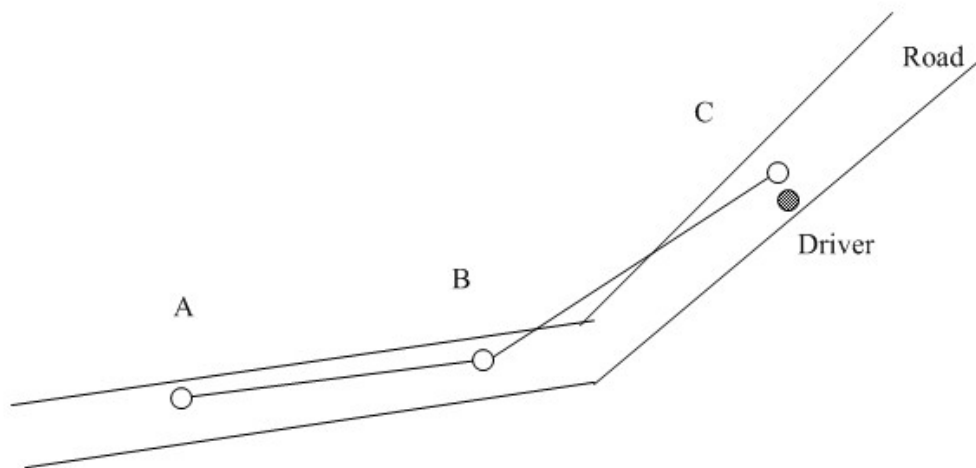


(algorithm_2)

3. When driver pass point A and move to point B, the algorithm also calculates distance between driver and point B (see algorithm_2). The system uses two float variables, “Old distance” and “New distance”. “Old distance” keeps last distance and “New distance” stores new distance. The algorithm uses them to check whether user driving on the speed zone. Here is a table to shows some data of driving.

ID	Old distance	New distance
1	200M	200M
2	200M	180M
3	180M	160M
4	160M	175M
5	175M	185M

In data 1, it is first time to get distance, so the old distance and new distance are same. From data 2 and 3, the new distance decreases from 180M to 160M. It means the driver moving to point B and driving near the speed route. However the new distance increases from 175M to 185M at data 4 and data 5, it shows the driver’s behavior is abnormal. Driver possible leaves speed zone.



(algorithm_3)

4. After pass point B, the algorithm uses same way to calculates relationship of driver and via point until arrive last point of this speed route. If the distance of driver and point C less than 5 meters, it means the driver is almost leave speed zone and the app begins to give a warning. This is a condition of driver leave speed zone.

Procedural of algorithm:

Begin

Calculate distance (d_1) between driver and start point of speed zone.

If distance (d_1) less than 5 meters

 Driver enter speed zone

 Display warning

 For each point in speed zone

 Calculate distance (d_n) between driver and point

 If old distance (d_{n-1}) bigger than new distance (d_n)

 User driving in the zone

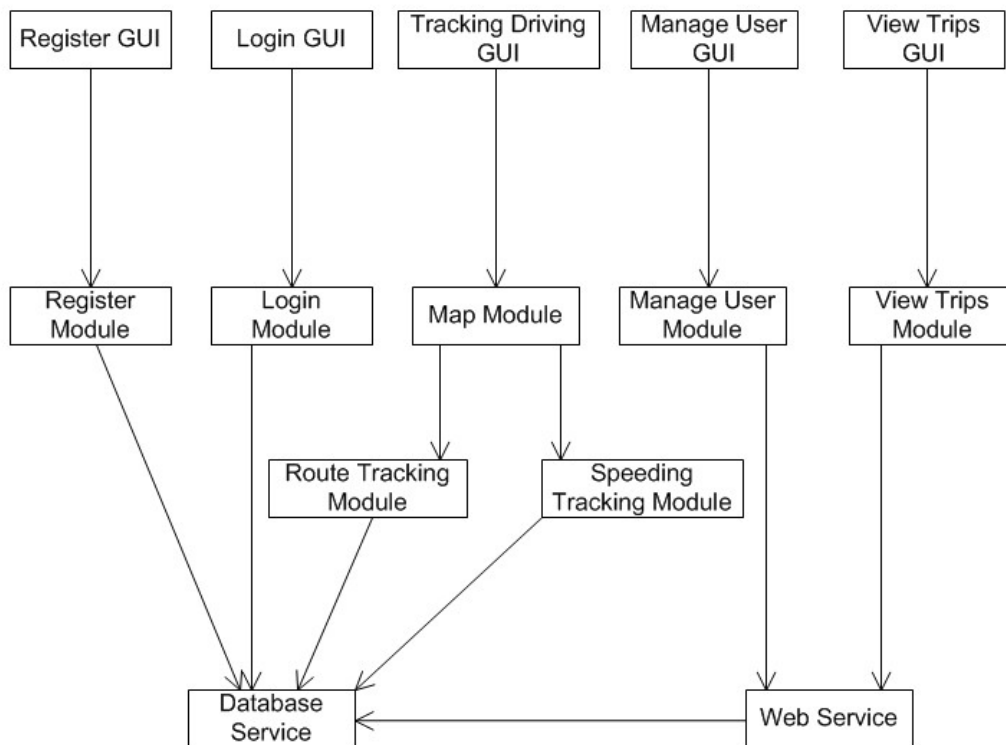
 Else

```

    Driver's behavior is abnormal
    Display warning
    Calculate distance (d_n) between driver and end point of speed zone.
    If distance (d_n) less than 5 meters
        Driver leave speed zone
        Display warning
    End
  
```

5. Module Design

5.1 Overview Dependencies between Modules



5.2 Brief Responsibilities of Major Module

- **Register Module**

This module is designed for users create new account. Module gets new account's data form Register GUI and all of data are stored into an array. After user touching 'OK' button, the new account be created in database.

Register GUI design (Android):



The image shows a screenshot of an Android application's 'Register Page'. The page has a black background with white text and input fields. The labels are 'Account name:', 'Password:', 'Confirm Password:', and 'E-mail address:'. At the bottom, there are two buttons: 'Cancel' on the left and 'OK' on the right.

(GUI-Register)

This picture shows register GUI design. The new users need to complete the detail of account, account name, password and e-mail address. Then they can choose to create it by touching 'Ok' or cancel by 'Cancel'

Data structures of Register account

This module data structures will be of an ARRAY. System stores new user account's detail into an array, after users click 'OK' button, the array's elements are inserted into database.

Related database table: tbl_user

Procedural of register module:

Begin

Get data from Register GUI

Store them into an array

Connect with database

Get new user's data from array

If 'OK' button is clicked then

 Insert data into database

 Display alter

End if

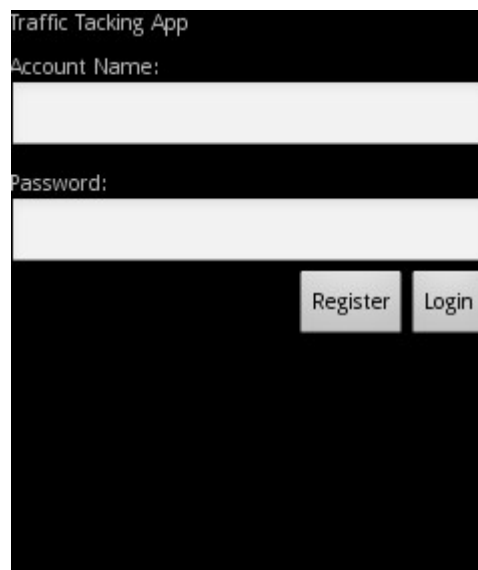
Close connect with database

End

● **Login Module**

This module is first step of start up. The aim of it is get identification of each user. Module gets user login's data (e.g. login name, password) from login GUI. When users click 'login' button, system selects username and password at database table.

User Login GUI design (Android)



(GUI-App Login)

This GUI is system login interface. Users input account name, password in text box and clicks 'login' button to run the app. They also can create new account by touching 'register' button.

Data Structures of Login

This module data structures also will be of an ARRAY. System stores user login's date (login ID and password) into an array. When users click 'login' button, system check username and password at database table, if they are correct, let user login, or else return warning information.

Related database table: tbl_user

Procedural of login module:

Begin

Get data from login GUI

Store them into an array

Connect with database

 Search data from user table

If username and password are correct then

Let user operation system

Else

Return error message

End if

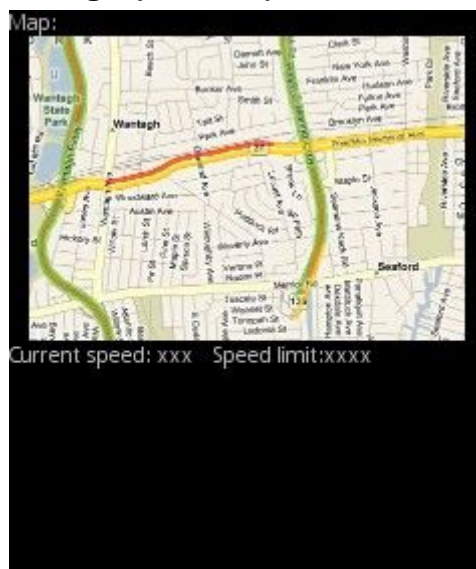
Close connect with database

End

● Map Module

This module is designed for displaying map information. Module gets driver's location and speed from GPS regularly and selects it on database, then the detail of routes (e.g. speed camera tracking routes) which around driver are shown on the map. It also can display user's speed.

Tracking Driving GUI design (Android)



(GUI-Tracking)

This screenshot is tracking driving GUI interface. Users can view their location, and the speed camera tracking route on the map. The speed limit and current speed are also shown under the map.

Data Structures of Map module

This module data structures will be of NVARCHAR and INTEGER. System keeps driver's location in NVARCHAR and sends it to database, then the detail of routes which around driver's location are shown on the map for viewing. System also stores driver's speed into INTEGER and displays to users.

Related database table: tbl_road, tbl_coords_road

Procedural of map module:

Begin

Get user's location

Connect with database

 Search location from road table and coords table

 If location is correct then

 Display route's information on the map

 Else

 Return error message

 End if

Close connect with database

End

● Route Tracking Module

This module is for app tracks user's driving route and stores them into database. It gets location/speed from map module frequently. After user finishes driving, module inserts these data into database, such as start point and end point of route, speed of user, major via points, etc.

Data Structures of Route tracking

This module data structures will be of NVARCHAR, INTEGER, LINESTRING and TIMESTAMP. System keeps start point and end point of route into NVARCHAR, stores max speed and average speed into INTEGER, fills some major via points into LINESTRING and uses TIMESTAMP to keep date of driving.

Related database table: tbl_driving_route

Procedural of route tracking module:

Begin

Get user's location from map module

Start tracking

Get data of driving route

End tracking

Connect with database

 Insert data into driving route table

 Display alter

Close connect with database

End

● Speeding Tracking Module

This module is designed for app tracks driver's speed and gives them a warning when over speed. Firstly module gets driver's location from map module, and then the area of enforcement zone which round driver is selected from database. The module matches coordinate of driver's location with speed zone's coordinate. If user driving in the zone, the app begins to speed tracking,

and if current speed is bigger than limit, app gives a warning to user. If users entry or leave enforcement zone, app also give a warning.

Data Structures of Speeding Tracking

This module data structures also will be of FLOAT, LINESTRING and INTEGER. Module stores coordinate of driver's location into FLOAT, speed into INTEGER and uses LINESTRING to keep location of speed trips.

Related database table: tbl_road, tbl_driving_road

Procedural of speeding tracking module:

Begin

Get user's location

Get current speed

Connect with database

 Get coordinates of speed zone and speed limit

 Use algorithm of speeding tracking

 Match the coordinate of zone with user's location

 If use entry the zone

 Display warning

 Begin tracking

 Match the current speed with speed limit

 If current speed is bigger than speed limit

 Display warning

 End if

 If user leave enforcement zone

 Display warning

 Stop tracking

 End if

End

- **Manage User Module**

This module is written for admin manages user's account. Module gets all of user's account data from database and sends them to web service. Admin can view and delete any user's account.

Data Structures of Manage users

This module data structures also will be of an ARRAY. System uses it to store users account's data from database, and then the website uses ARRAY to display user's information to manager.

Related database table: tbl_user

Manage User GUI design (PC)

Manage Users					
ID	AccountName	Passwrod	RegisterTime	E-mail Address	
1	Tome	123	12-12-2010	tome@gmail	Delete
2	Jake	2w3e	10-2-2009	jake@gmail	Delete
...

(GUI-Manage Users)

The GUI-Manage Users is designed for admin. The administrator manages user's account on the website. It has account ID, account name, password, register time etc. Any of account can be deleted by clicking 'Delete' button.

Procedural of manage user module:

Begin

Connect with database

Select user's account from user table
 If "delete" button is clicked
 Delete this account from user table
 End if
Display alter
Close connect with database
End

- **View Trips Module**


This module is designed for users check driver's behavior from internet. Module select user's driving route from database and sends then to web service. Users can get detail of each trip. (E.g. user name, data of trip, max speed, route of trip.) If they click 'view' button, this trip can be shown on the map.

Data Structures of View trips

This module data structures will be of an ARRAY. System fills it with route's data which comes from database, such as start location, end location, speed, date, etc. Then this ARRAY is used for displaying driving trips on the web site.

Related database table: tbl_route

View Trips GUI design (PC)

Traffic data						
ID	Name	Date	Max Speed	AverageSpeed	Route of Trip	View detail on map
1	Tome	6-12-2010	120Mph/h	80Mph/h		<input type="button" value="View"/>
...

(GUI-View Trips)

This screenshot shows how the system displays the data of trips to users. There is a table on the webpage which includes ID, Name, Date, Max Speed, etc. Users also can quickly view their trip's detail on map by clicking 'View' button.

Procedural of view trips module:

Begin

Connect with database

 Select user's trips from route table

 Display detail of them on web site

 If "view" button is clicked

 Display route on the map

 End if

Close connect with database

End

6. Conclusions

This document is general introduction design of Traffic Tracking Application on Android. It includes data structure design, architectural design, module design, interface design and procedural design. This document is very important and it can promote efficiency during programming. Coding and testing should be following with this document.

Reference:

[1] SpatiaLite 2.3.1, Alessandro Furieri, From SpatiaLite

<http://www.gaia-gis.it/spatialite/> [accessed 18-1-2011]

[2] Map of safety cameras, From Ireland's National Police Service

<http://www.garda.ie/GoSafe.htm> [accessed 23-1-2011]

[3] Java API for XML Processing Tutorial, From Java.sun.com

<http://java.sun.com/webservices/reference/tutorials/jaxp/html/intro.html>

[accessed 24-1-2011]