APPENDIX-I Modules of Traffic Alert System and their Pseudo-code listing following Figure 1 in document.

----------------------------------------------------------

Start-Up subsystem & Sub-Modules of the System as was shown in Figure 1 inside the text.

----------------------------------------------------------

//\*\*\*\*\*\*\* Main method : Startup procedure. \*\*\*\*\*\*\*\*//

/\*\* The program begins from this module. The Startup module initializes all the top-level modules of the system. These top-level modules in turn declare and define the required variables and sub-modules.\*/

Startup procedure

{

- Start the web-server on the central server machine

- invoke "ENVIRONMENT SUBSYSTEM"

- invoke "MANAGEMENT SUBSYSTEM"

- invoke "DETECTION SUBSYSTEM"

- invoke "RESPONSE SUBSYSTEM"

}

-------------------------------------------------------------

Top level modules

-------------------------------------------------------------

/\*The Environment Subsystemuses MySQL as the datastore. It defines the tables required to store user login, subscribers, geo-location, mobile numbers and 3 degrees of alerts of the generated traffic map. This module creates a pool of database connections that can be re-used by the system. It defines sub-modules to query and insert records in to the database tables.\*/

Environment Subsystem{

- define "Connection Pool" function

- invoke "Create Database" function

- invoke "Initialize Server Database" function

- invoke "Initialize Client Database" function

{

SERVER DATABASE sub-module

{

* Utilizes "Connection Pool" function
* Utilizes "Create Table" function

}

CLIENT DATABASE sub-module

{

* Utilizes "Connection Pool" function
* Utilizes "Create Table" function

}

}

}

/\*The **MANAGEMENT SUBSYSTEM** module defines sub-modules to create GUI. It provides a HTML page for user login to enter the system. It downloads the GUI to the client machine on successful user login. This module provides controls to: view the traffic maps from the integrated traffic evaluation system and alert system, and send warning SMS and to create new subscribers. \*/

MANAGEMENT SUBSYSTEM {

- define "Initialize GUI" sub-module

- define "Create New Subscriber" sub-module

- invoke "User Login" sub-module

}

/\*The **Detection subsystem** receives the encapsulated information packet from the Management Subsystem. Based on the level of congestion/alert/event, it incorporates the frequency of SMS. It then triggers the Response subsystem to send warning SMSs to the mobile numbers in the affected zone.\*/

Detection Subsystem

{

- declare and set the ALERT\_TO\_SMS\_FREQUENCY\_MAPPING variable.

/\*This contains the mapping between Congestion level and Frequency of SMS \*/

- define "Process Information Packet" sub-module

- define "Searching and Matching" sub-module

- define "Set SMS Frequency" sub-module

- define "Trigger Alert" sub-module

}

/\*The **Response Subsystem** receives the information packets from the Detection subsystem and performs the actual delivery of the warning messages to subscriber mobile numbers. The delivery of messages is logged and shown to user on the client GUI.

\*/

Response Subsystem

{

}

----------------------------------------------------------------

Sub-Modules of ENVIRONMENT SUBSYSTEM.

----------------------------------------------------------------

Create Database

{

- create the "Location" table with the following fields:

pixel value in the generated traffic geo-map

zone (latitude/longitude)

congest/alert level (LOW, MODERATE, HIGH)

- create the "Alert Message" table with the following fields:

zone

level

alert message

- create the "Subscriber" table with the following fields:

subscriber name

subscriber mobile number

zone

- set appropriate permissions to the tables

- invoke "Create Connection Pool" sub-module of "Database module"

}

Initialize Server Database

{

}

Initialize Client Database

{

}

Connection Pool

{

}

Create Connection Pool

{

}

}

Get Database Connection

{

- if CONNECTION\_POOL is not empty

{

- return a database connection from the CONNECTION\_POOL

}

else

{

- return an error indicating that CONNECTION\_POOL is empty

}

}

Release Database Connection

{

- release all resources related to the database connection

- return the connection to the CONNECTION\_POOL

}

Get Zone for Pixel

{

}

Get Subscriber Data

{

}

Get Location Alert Level

{

}

Get Alert Message

{

- receive the alert level as input

- invoke "Get Database Connection" sub-module of "Database module"

- query the alert message associated with the congestion level from the

"Alert Message" table

- invoke "Release Database Connection" sub-module of "Database module"

- return the retrieved alert message

)

Insert Subscriber Data

{

- receive the following input parameters:

subscriber name

mobile number

zone (latitude/longitude)

- validate the input parameters

- invoke "Get Database Connection" sub-module of "Database module"

- insert a subscriber record into the "Subscriber" table

- invoke "Release Database Connection" sub-module of "Database module"

- return the status of the transaction : success or failure

}

Get Password for Username

{

- invoke "Get Database Connection" sub-module of "Database module"

- invoke "Release Database Connection" sub-module of "Database module"

}

Store Geo-Location Congestion Level

{

}

----------------------------------------------------------------

Sub-Modules of MANAGEMENT SUBSYSTEM.

----------------------------------------------------------------

/\*The Traffic-Alert-System can be implemented over the web using a URL. Once the URL is entered in a browser, an initial login HTML page is sent from the server and displayed to user. Here, user provides the login credentials and submits the request. The login request is sent to the server for authentication. If the authentication succeeds, the client GUI of the Traffic-Alert-System downloads to the user's machine from the server.\*\*/

User Login

{

- until the user successfully logs in to the system or the browser window is

closed, execute the following steps

{

}

}

}

Authenticate User

{

}

Initialize GUI

{

}

else if BUTTON\_CLICKED is Send SMS

{

- invoke "Create Confirmation Dialog" sub-module of

"Management Subsystem" with the following message:

"SMS traffic alert will be sent to all mobile numbers in the database

in the given map area. This may take a while. Do you wish to

continue?"

- if "YES" status is returned in the previous step

{

- for every pixel in the output traffic map execute the

following steps:

{

- invoke "Get Zone for Pixel" sub-module of the

"Database module" by passing the pixel value from the

output traffic map

- invoke "Get Subscriber Data" sub-module of the

"Database module" by passing the zone retrieved in the

previous step

- create an information packet with the following

information:

geo-location (latitude/longitude)

list of mobile numbers present in this geo-location

- invoke "Process Information Packet" sub-module of

"Detection Subsystem" and pass the information packet created in the previous step to it.

}

}

}

Display Map

{

}

Process Maps

{

Create New Subscriber

{

}

Write to UI Log

{

}

----------------------------------------------------------------

Sub-Modules of Detection Subsystem.

----------------------------------------------------------------

Process Information Packet

{

}

Get SMS Frequency

{

- return the SMS Frequency

}

Trigger Congestion Warning

{

a string mentioning that the SMS sending is triggered

"Log" tab

}

Searching and Matching

{- algorithm for string searching

- sequential order context searching mode

- exact string through complete matching

- forward chaining inferencing rules apply

}

----------------------------------------------------------------

Sub-Modules of Response subsystem.

----------------------------------------------------------------

Send SMS

{ - accept the subscriber mobile numbers, zone, associated alert congestion message

and SMS frequency as input

- validate the input parameters

- Initialize ROUND\_NUMBER variable to 1

- Process the following steps while ROUND\_NUMBER is less than or equal to

"SMS frequency"

{

- display the ROUND\_NUMBER of the SMS being sent in the client GUI

- if DELIVERY\_MODE is CBC

{

- call "Send SMS CBC" sub-module

}

else if DELIVERY\_MODE is SELF\_HARDWARE

{

- call "Send SMS SELF\_HARDWARE" sub-module

}

else if DELIVERY\_MODE is INTERNET\_SOFTWARE

{

- call "Send SMS INTERNET\_SOFTWARE" sub-module

}

else

{- DELIVERY\_MODE is not specified, display error message to

user

}

- display the status of SMS sent in client GUI

- increment ROUND\_NUMBER variable by 1

}

}

/\*\*The advantage of the CBC system is that it allows the sending of messages without having to know the phone numbers of the users in the region. Instead of sending a message to a specific known mobile one you can send a text to all mobile phones in a specific zone.\*\*/

Send SMS CBC

{

}

/\*\*This required the system to be connected to a SMS gateway and a modem that can send SMSs\*\*/

Send SMS SELF\_HARDWARE

{

}

/\*\*This uses websites on the internet that provide free SMS sending capabilities. The administrator needs to pre-register on such websites to use the SMS sending capabilities.\*/

Send SMS INTERNET\_SOFTWARE