FRIX-Traffic Analyzer And Transportation Assistant

Manuj Darbari, Prateek Kumar Singh,Rakesh Kumar,Sameer Eshan
Department of Information Technology,
Babu Banarasi Das National Institute of Technology,
Lucknow, India
manujuma@rediffmail.com

Savitur Prakash
Lucknow Doordarshan Kendra,
Lucknow, India

saviturprakash@rediffmail.com

Abstract— **This website specializes in giving traffic reports for a specific area. Much of our collected data comes from automatic sensors placed at various points around the motorway to monitor vehicle speeds. We have provided an accurate, up-to-date report which is normally faster than other mediums, such as radio. Typically our speed readings are updated every 3 minutes.**
Using this website a user can give the starting place and destination to the system. Depending on the prevalent traffic conditions, the website will suggest the best route to be taken by the user. Other than this, the user can also generate several other useful information from the data collected, like finding the nearest hospital to the user's location or finding the nearest restaurant. There is also a scope to add lot more to this application for user's convenience.

Keywords-component; formatting; style; styling; insert (key words)

I. INTRODUCTION

The purpose of this project is to provide the user intelligent transportation system (ITS)[1,2,4] which refers to the efforts to add information and communications technology to transport infrastructure and vehicles in an effort to manage factors that typically are at odds with each other, such as vehicles, loads, and routes to improve safety and reduce vehicle wear, transportation times, and fuel consumption.

II. FEATURES OF THE SYSTEM

- Generates the traffic report of certain areas.
- Provides the user with the best path from one location to another after analysing distance and traffic condition between two places.
- Provides the use with the distance, time required to travel, and direction from one place to another entered by the user.
- Gives user the ability to plan his travel and to choose his route according to his convenience.
- Gives the user information regarding the road blockage, traffic jams and the areas of slow traffic using markers on the maps.
- Search function is provided to the user to find the location of any place or landmark on the map using a marker.
- Gives the user an option to search the road and check the traffic conditions on the same.

A. Performing Charecterstics

The following performance characteristics were taken care of in developing the systems:

- User Friendliness: The system is easy to learn and understand. A naive user can also use the system effectively, without any difficulty.
- User Satisfaction: The system is such that it stands up to the user's expectations and requirements.
- Response Time: The response time of all the operations is very high. This has been made possible by careful programming.
- Error handling: Response to user errors and undesired situations have been taken care of to ensure that the system operates without halting in case of such situation and proper error messages are given to user.
- Safety: The system is able to avoid catastrophic behavior.

III. USER CHARACTARISTICS

- All users must have a little experience on working on computer and have basic idea of net browsing.
- All users should know basic English.

WORKING OF THE TRAFFIC ANALYSER

Google created the Google Maps API to allow developers to integrate Google Maps into their websites with their own data points. It is a free service, and currently does not contain ads, but Google states in their terms of use that they reserve the right to display ads in the future.

By using the Google Maps API, it is possible to embed the full Google Maps site into an external website. Developers are required to request an API key, which is bound to the website and directory entered when creating the key. The Google Maps API key is no longer required for API version 3. Creating a customized map interface requires adding the Google JavaScript code to a page, and then using Javascript functions to add points to the map.
When the API first launched, it lacked the ability to geocode addresses, requiring users to manually add points in (latitude, longitude) format. This feature has since been added for premier.
At the same time as the release of the Google Maps API, Yahoo! released its own Maps API. Yahoo! Maps, which lacks international support, included a geocoder in the first release.
As of October 2006, the implementation of Google Gadgets' Google Maps is simpler, requiring only one line of script, but it is not as customizable as the full API
The fundamental element in any Google Maps API application is the "map" itself. This document discusses usage of the fundamental GMap2 object and the basics of map operations.
Loading the Google Maps API.

```
<script
src="http://maps.google.com/maps?file=
api&v=2&key=abcdefg&sensor=true_or_fal
se"
        type="text/javascript">
</script>
```

The
http://maps.google.com/maps?file=api&v=2&key=abcdefg
URL points to the location of the JavaScript file that includes all of the symbols and definitions you need for using the Google Maps API. Your page must contain a script tag pointing to this URL, using the key you received when you signed up for the API. In this example the key is shown as "abcdefg."
Note that we also pass a sensor parameter to indicate whether this application uses a sensor to determine the user's location. We've left this example as a variable *true_or_false* to emphasize that you **must** set this value to either true or false explicitly.

Map DOM Elements

```
<div id="map_canvas" style="width:
500px; height: 300px"></div>
```

For the map to display on a web page, we must reserve a spot for it. Commonly, we do this by creating a named div element and obtaining a reference to this element in the browser's document object model (DOM).
In the example above, we define a div named "map_canvas" and set its size using style attributes. Unless you specify a size explicitly for the map using GMapOptions in the constructor, the map implicitly uses the size of the container to size itself.

GMap2 - the Elementary Object

The JavaScript class that represents a map is the GMap2 class.

```
var map = new
GMap2(document.getElementById("map_canv
as"));
```

Objects of this class define a single map on a page. (You may create more than one instance of this class - each object will define a separate map on the page.) We create a new instance of this class using the JavaScript new operator.
When you create a new map instance, you specify a DOM node in the page (usually a div element) as a container for the map. HTML nodes are children of the JavaScript document object, and we obtain a reference to this element via the document.getElementById() method.
This code defines a variable (named map) and assigns that variable to a new GMap2 object. The function GMap2() is known as a *constructor* and its definition is shown below:

Constructor	**Description**
GMap2(container, opts?)	Creates a new map inside the given HTML *container*, which is typically a DIV element. You may also pass optional parameters of type GMap2Options in the *opts* parameter.

Table.1 Constructor definition of Gmap2

Note that because JavaScript is a loosely typed language, we do not need to pass any optional parameters in the constructor, and we don't do so here.
Initializing the Map

```
  map.setCenter(new GLatLng(37.4419, -
122.1419), 13);
  map.setUIToDefault();
```

Once we've created a map via the GMap2 constructor, we need to initialize it. This initialization is accomplished with use of the map's setCenter() method. The setCenter() method requires a GLatLng coordinate and a zoom level and this method **must** be sent before any other operations are performed on the map, including setting any other attributes of the map itself.

B. Loading the Map

Additionally, we also call `setUIToDefault()` on the `map`. This method sets up the map's user interface (input handling and set of controls) to a default configuration, including pan and zoom controls, selection of map types, etc.

```
<body onload="initialize()"
onunload="GUnload()">
```

While an HTML page renders, the document object model (DOM) is built out, and any external images and scripts are received and incorporated into the document object. To ensure that our map is only placed on the page after the page has fully loaded, we only execute the function which constructs the GMap2 object once the <body> element of the HTML page receives an onload event. Doing so avoids unpredictable behavior and gives us more control on how and when the map draws.

The onload attribute is an example of an event handler. The Google Maps API also provides a number of events that you can "listen" for to determine state changes. The GUnload() function is a utility function designed to prevent memory leaks

B. Latitudes and Longitudes

Now that we have a map, we need a way to refer to locations on the map. The GLatLng object provides such a mechanism within the Google Maps API. You construct a GLatLng object, passing its parameters in the order { latitude, longitude } as is customary in cartography:

```
 var myGeographicCoordinates = new
GLatLng(myLatitude, myLongitude)
```

Just as it is useful to easily refer to a geographic point, it is also useful to define the geographic bounds of an object. For example, a map displays a current "window" of the entire world within what is known as a viewport. This viewport can be defined by the rectangular points at its corners. The GLatLngBounds object provides this functionality, defining a rectangular region using two GLatLng objects representing the southwest and northeast corners of the bounding box, respectively.

GLatLng objects have many uses within the Google Maps API. The GMarker object takes a GLatLng in its constructor, for example, and places a marker *overlay* on the map at the given geographic location.

Map Attributes

Each map contains a number of attributes that may be inspected or set. For example, to find the dimensions of the current viewport, use the GMap2 object's getBounds() method to return a GLatLngBounds value.

Each map also contains a *zoom level*, which defines the resolution of the current view. Zoom levels between 0 (the lowest zoom level, in which the entire world can be seen on one map) to 19 (the highest zoom level, down to individual buildings) are possible within the normal maps view. Zoom levels vary depending on where in the world you're looking, as data in some parts of the globe is more defined than in others. Zoom levels up to 20 are possible within satellite view.

You can retrieve the current zoom level in use by the map by using the GMap2 object's getZoom() method.

C. Map Interactions

Now that you have a `GMap2` object, you can interact with it. The basic map object looks and behaves a lot like the map you interact with on the Google Maps website and comes with a lot of built-in behavior. The `GMap2` object also provides a number of configuration methods to alter the behavior of the map object itself.

By default, map objects tend to react to user activity as they do on `http://maps.google.com`. You can alter this behavior with a number of utility methods, however. For example, the `GMap2.disableDragging()` method disables the ability to click and drag the map to a new location.

You can also interact with the map programmatically. The `GMap2` object supports a number of methods that alter the map state directly. For example, the `setCenter()`, `panTo`, and `zoomIn()` methods operate on the map programatically, rather than through user interaction.

The following example displays a map and provides a button to initiate a `panTo` method, which centers the map at a given point. If the specified point is in the visible part of the map, then the map pans smoothly to the point; if not, the map jumps to the point.

```
  var map;

  function initialize() {
    if (GBrowserIsCompatible()) {
      map = new
GMap2(document.getElementById("map_canv
as"));
      map.setCenter(new
GLatLng(37.4419, -122.1419), 13);
    }
  }

  function animate() {
```

D. JAVA SCRIT GENERATION

Java script is an object-oriented language that allows creation of interactive web page. Java script allows user entries, which are loaded into an HTML form to be processed as required. This empowers a web to return site information according to user's requests.

Java script offers several advantages to a web developer, a short development cycle, Easy to learn, Small size scripts. The strength of java script can be easily and quickly used to extend the functionally of HTML pages, already on a web site.

THE ADVANTAGES OF JAVA SCRIPT

An interpreted language

Java script is an interpreted language, which requires no compilation steps. This provides an easy development process. The syntax is completely interpreted by the browser just as it interpreted HTML tags.

Embedded within HTML

Java script does require any special or separate editor for programs to be written, edited or compiled. It can be written in any text editor like notepad, along with appropriate HTML tags, and saved as filename.html. HTML files with embedded JavaScript commands can be read and interpreted by any browser that is JavaScript enabled.

Minimal syntax-easy to learn

By learning just a few commands and simple rules of syntax. Complete application can be built using JavaScript.

Performance

Java script can be written such that the HTML files are fairly compact and quite small. This minimizes storage requirement on the web server and download time for the client.

Procedural Capability

Very programming language needs to support facilities such as conditions checking, looping and branching. Java script provides syntax, which can be used to add such procedural capabilities to web page coding.

Designed For Programming User Events

Java scripts support object/events based programming.

Easy Developing and Testing

Being an interpreted language, JavaScript scripts are tested line by line, and errors are also listed as they are encountered, i.e. an appropriate error message along with line number is listed for every error that is encountered. It is thus easy to locate errors, make changes, and text it again without the overhead and delay of compiling.

Platform Independence/Architecture Neutral

JavaScript is a programming language that is completely independent of the hardware on which it works. It is a language that is understood by any JavaScript enabled browser. Thus, JavaScript applications work on any machine that has a JavaScript enabled browser installed. This machine can be any where on the network.

Writing JavaScript into HTML

JavaScript syntax is embedded into an HTML file. A browser reads HTML files and interprets HTML tags. Since all JavaScript need to be includes as an integral part of an HTML document when required, the browser need to be informed that specific sections of HTML code is JavaScript. The browser will then use it's built in JavaScript engine to interpret this code.

The browser is given this information using the HTML tags <SCRIPT>...</SCRIPT>. The <SCRIPT> tag marks the beginning of JavaScript.

A snippet beginning of scripting code. The paired <SCRIPT> marks the end of the snippet of scripting code.

Like most other HTML tags, the <SCRIPT> tags takes in an optional attribute, as listed below:

Language: Indicates the scripting language used for writing the snippet of scripting code.

SYNTAX:

```
<SCRIPT LANGUAGE="JavaScript">
        JavaScript code snippet written here
  </SCRIPT>
```

E. Logic Used:

In this web application the routes are treated as a directed graph. Each edge of the graph is assigned a weight depending upon the distance and traffic conditions of that path so **Dijkstra algorithm** is used to find the best path.

Dijkstra Algorithm:

Let's call the node we are starting with an initial node. Let a distance of a node Y be the distance from the initial node to it. Dijkstra's algorithm will assign some initial distance values and will try to improve them step-by-step.

1. Assign to every node a distance value. Set it to zero for our initial node and to infinity for all other nodes.

2. Mark all nodes as unvisited. Set initial node as current.

3. For current node, consider all its unvisited neighbors and calculate their distance (from the initial node). For example, if current node (A) has distance of 6, and an edge connecting it with another node (B) is 2, the distance to B through A will be 6+2=8. If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.

4. When we are done considering all neighbors of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.

5. Set the unvisited node with the smallest distance (from the initial node) as the next "current node" and continue from step 3.

```
1   function Dijkstra(Graph, source):
2   for each vertex v in Graph:          // Initializations
3    dist[v] := infinity                 // Unknown distance function from source to v
4    previous[v] := undefined            // Previous node in optimal path from source
5   dist[source] := 0                    // Distance from source to source
6    Q := the set of all nodes in Graph  // All nodes in the graph are unoptimized - thus are in Q
7   while Q is not empty:                // The main loop
8   u := vertex in Q with smallest dist[]
9   if dist[u] = infinity:
10  break  // all remaining vertices are inaccessible
11  remove u from Q
12  for each neighbor v of u: // where v has not yet been removed from Q.
13  alt := dist[u] + dist_between(u, v)
14  if alt < dist[v]:  // Relax (u,v,a)
15  dist[v] := alt
16  previous[v] := u
17  return previous[]
```

An upper bound of the running time of Dijkstra's algorithm on a graph with edges E and vertices V can be expressed as a function of |E| and |V| using the Big-O notation.
For any implementation of set Q the running time is O(|E|*decrease_key_in_Q + |V|*extract_minimum_in_Q), where decrease_key_in_Q and extract_minimum_in_Q are times needed to perform that operation in set Q.

The simplest implementation of the Dijkstra's algorithm stores vertices of set Q in an ordinary linked list or array, and operation Extract-Min(Q) is simply a linear search through all vertices in Q. In this case, the running time is O(|V|2+|E|)=O(|V|2).
For sparse graphs, that is, graphs with fewer than |V|2 edges, Dijkstra's algorithm can be implemented more efficiently by storing the graph in the form of adjacency lists and using a binary heap, pairing heap, or Fibonacci heap as a priority queue to implement the Extract-Min function efficiently. With a binary heap, the algorithm requires O((|E|+|V|) log |V|) time (which is dominated by O(|E| log |V|), assuming the graph is connected), and the Fibonacci heap improves this to O(| E | + | V | log | V |).

F. DATABASE STRUCTURE:

The database frix.db have two tables:

1. Places
2. Distance

The Places Table as shown in Table 2 consists of all the relevant fields like Place Id, Name of the Place, Latitude, Longitude and a brief description about that place.

S.No.	Name of Field	Data type
1	Id	Int (auto increment)
2	Name	Varchar(200)
3	Latitude	Double
4	Longtitude	double
5	Description	Varchar(2000)

Table 2. Places description Table

The Distance Table shown in Table 3 consists of all the fields related to the Id of two places whose distance have to be measured and the time it will take to reach the location Id 2

S.No.	Name of Field	Data type
1	Id1	Int
2	Id2	Int
3	Distance	Float
4	Time	Float

Table 3. Distance Measurement Table

IV. ANALYSIS OF TRAFFIC ANALYSER

Intelligent RFID traffic control[3,5,6,7] has been developed for dynamic traffic light sequence. It has circumvented or avoided the problems that usually arise with systems such as those, which use image processing and beam interruption techniques. RFID technology with appropriate algorithm and data base were applied to a multi vehicle, multi lane and multi road junction area to provide an efficient time management scheme. A dynamic time schedule was worked out for the passage of each column. The simulation has shown that, the dynamic sequence algorithm has the ability to intelligently adjust itself even with the presence of some extreme cases. The real time operation of the system emulates the judgment of a traffic policeman on duty, by considering the number of vehicles in each column and the routing proprieties.

The website specializes in giving traffic reports for a specific area. Much of our collected data comes from automatic sensors placed at various points around the motorway to monitor vehicle speeds. This means we can provide an accurate, up-to-date report which is normally faster than other mediums with readings being updated every 3 minutes.
Using this website a user can give the starting place and destination to the system. Depending on the prevalent traffic conditions, the website will suggest the best route to be taken by the user.

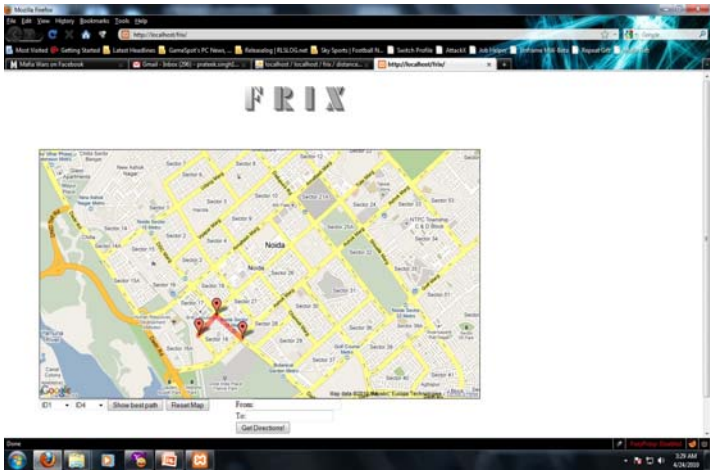

Figure 1. Plot showing best path from ID1to ID 4

The website suggests the best possible path between two destinations ID1 to ID 4 it can also perform the local search on the map by giving the details about that location a shown in the figure 2.

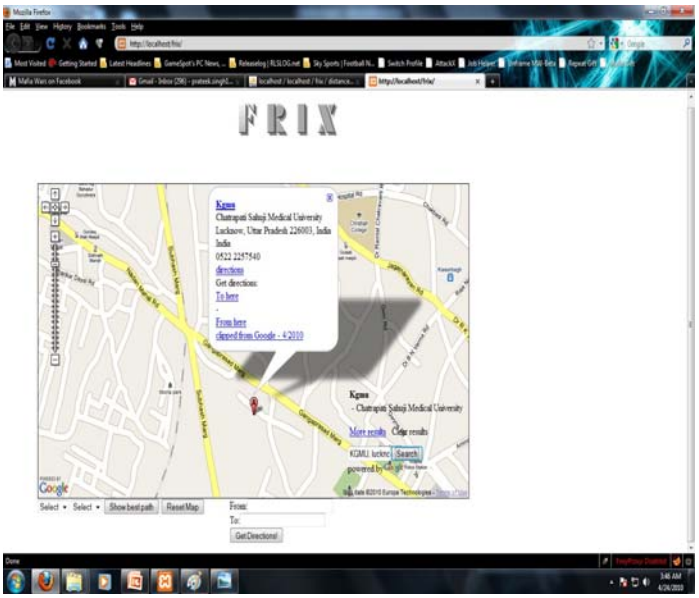

Figure 2. Application performing the local search on the map

If a user wants to find out the condition of traffic at a particular location he is upgraded with the information about a particular road accident that has happened as result there is a heavy jam condition at a particular location as shown in figure 3.

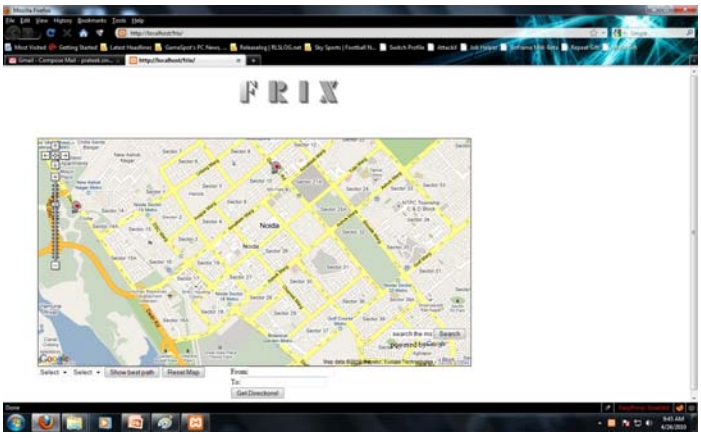

Figure 3. Markers in the map showing the roads incidents on map causing the traffic jam

V. CONCLUSION

Overall the traffic analyzer provides complete information about a particular location. By using Dijkstra algorithm the user is able to find the optimum path and its distance; it also helps the commuter in finding out the exact condition of the traffic at a particular location or between two locations.
In future we will be upgrading our system with stochastic mapping of the FRIX we will emulate close to real time traffic movement.

REFERENCES

[1]. Di Febbraro, A., & Sacco, N. (2004). On modeling urban transportation networks via hybrid Petri nets. *Control Engineering Practice*, *12*(10), 1225-1239.
[2]. Di Febbraro, A., & Sacco, N. (2004). An urban traffic control structure based on hybrid Petri nets. *IEEE Transactionson Intelligent Transportation Systems*, *5*(4), 24-237.
[3]. Di Febbraro, A., Giglio, D., & Sacco, N. (2002). On applying Petri nets to determine optimal offsets for coordinated traffic light timings. *Proceedings of the 5th IEEE International Conference on Intelligent Transportation Systems*, Singapore (pp. 87-706).
[4]. Gallego, J.-L., Farges, J.-L., & Henry, J.-J. (1996). Design by Petri nets of an intersection signal controller. *Transportation Research Part C*, *4*(4), 231-248.
[5]. List, G. F., & Cetin, M. (2004). Modeling traffic signal control using Petri nets. *IEEE Transactions on Intelligent Transportation Systems*, *5*(3), 177-187.
[6]. Jensen, K. (1992). *Colored Petri nets: basic concepts, analysis methods and practical use*, Vol. 1. New York: Springer.
[7]. List, G. F., & Cetin, M. (2004). Modeling traffic signal control using Petri nets. *IEEE Transactions on Intelligent Transportation Systems*, *5*(3), 177-187.
[8]. Tzes, A., Kim, S., & McShane, W. R. (1996). Applications of Petri networks to transportation network modeling. *IEEE Transactions on Vehicular Technology*, *45*(2), 391-400.
[9]. Google Api playground

AUTHORS PROFILE

Manuj Darbari is working as Associate Professor in Department of Information Technology, Babu Banarasi Das National Institute of Technology, Lucknow. He has guided several masters students and guiding two doctoral thesis in the field of MIS and Soft computing. He has published more than twenty five research papaers in national and international journal. He is also reviwer of many international journals.

Prateek Kumar Singh,Rakesh Kumar,Sameer Eshan are students of undergraduate program of BBDNITm,Lucknow.

Savitur Prakash is workimg as senior design engineer at lucknow doordarshan his area of interest include digital image processing and soft computing.