Get Direction: Optimal Routes for Travelling with respect to Time and Cost

Mrs. Sunantha Krishnan, Kevin D'souza, Neville D'souza, Abhinav Singh and Sharad Bhatt
Department of Information Technology
Don Bosco Institute of Technology
University of Mumbai
Mumbai, India
(sunanthag1998, kdsonik.11, bhatt.sh2 and abhinav.thegame,nevilledsouza28) @gmail.com

Abstract*: Optimization of time and money spent on travelling is an issue faced by everyone. Get Direction is a traveler's guide web page to get you directions and to plan tour and travel, with help of Google map API as base. The user will we asked to select details like mode of transport, timing and source destination. So this webpage will display various ways to reach destination with different of modes travel, the associated duration and cost. The scope is using Google map to calculate the distance of travel, displaying the resulting travel route with break of journey and mining system database to determine the associated cost. The system utilizes a knowledge base formed by tracking user's action. Also the system provides ticketing feature where the user is asked to enter his credit card details for payment.

Keywords – .Net, SQL, Google Maps API, AJAX, JavaScript.

I. INTRODUCTION

In a city like Mumbai people travel a lot and usually need information on the travel route in case of driving and various transit options available for making the travel and also the time and cost associated for the travel. Many would ask their friends for advice or would take help of information available with brochures, tourist guides and different web pages. The challenge is to present all this data together in one system. Thus developing a system which will enable people to navigate from Source A to Destination B by providing them to an option to choose their mode of transport i.e. transit (buses and trains) or driving (rickshaw, cab or a private vehicle) displaying alternative routes, stating the time and money for the travel. Get Direction is using Google API which is used for getting optimal paths between source and destination and provides traveling information to users over the web by use of online Google maps. The benefit brought by the presence of this system is not only to inform users about the optimal paths between two places but provide them with associated with that path. Get Direction targets both tourists that are interested in travel information during their visit to a particular city, and also citizens that want to discover hidden secrets of their own city.

II. PROBLEM STATEMENT

This project is about providing a web service which works on any operating system and is aimed at providing a better solution than the present web based navigational applications. The basic idea is to give the user a schedule planned with the optimal travel routes and the cost for that travel. The idea revolves around extracting information using Google maps API and the web systems database. Thus a code to extract information for Google API, a database maintaining a collection of records, a web server hosting the web pages are some of the requirements for this system. "**GET DIRECTION**" is intended to have following features:

- The system provides the user with an option to choose the mode of transport.
- The system provides the user with optimal travel routes based on the selected mode of transport.
- The system provides the user, the cost associated with each of the route selected. In case of driving, information regarding fuel cost can be calculated. In case of transit mode, the cost of the bus and train ticket is provided for the selected travel route and also user is allowed to book tickets.
- The system tracks the user's action, and also the various transactions the user has made.
- A user friendly interface and runs on any given operating system.

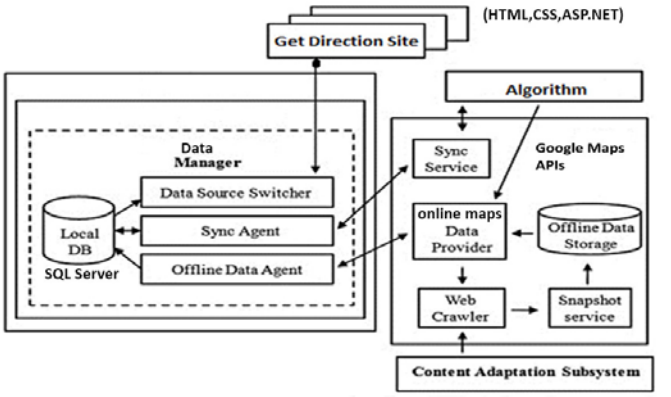
III. SCOPE

The project has a wide scope as it is assessable from any web enabled system and provides people with information which

would prove useful for travelling to various places in the city using both public as well as private means of transport.We have designed an application that will provide users with optimal travel routes where the user can see the selected route path on a map along with overlays that will help the user differentiate the objects used in a map such as highways, railway station, bus stops and then the cost associated with each of the route is displayed. In case of driving, fuel cost can be calculated whereas in transit mode, the cost of the bus and train ticket is provided for the selected travel route and the user is allowed to book tickets where the users credit card and other personal details are acquired and validated. The system tracks the user's action, and also the various transactions the user has made.

IV. DESIGN

System Architecture:



System Architecture (figure 1)

The above diagram is a pictorial description of the System Architecture. The figure shows 3 components namely: The get direction website, Server Side and the Google Map API.

Get Direction site: It is being developed in .NET as .NET has a much flexible architecture over PHP which doesn't allow us to use many other web technologies like AJAX. It is mainly concerned with client interaction, where user creates his account and selects the mode of transport he wish to find information about also interacts with the server of "Get Direction" and passes the values such as user details, source, destination and time.

The server side: It contains a database which stores this information and SQL query for sending and retrieving data from the Google map API. The Google Map API consists of many Javascript functions but the site will be using only a few functions such as 'Direction Rendering'.

The Local DB Server: It is an SQL Server. The Database contains all the information regarding the locations of the places through which the routes will be generated. The Database will also contain the user's personal information, account information and also the user's history can also be tracked and stored in the database.

The Google map API: It provides optimal travel routes and also travels information of each of these routes. The retrieved information is linked to the local database which provides us with additional information of cost associated with that travel along with the optimal paths between them.

The Web crawler: It is used to reduce the load time of the website by retrieving through the selected locations efficiently. The sync service is responsible for maintain the client and server. The Content Adaptation Subsystem: It is a lightweight HTTP like protocol which is used to extend transparent proxy servers which then frees up the clients resources and improves the systems performance. The snapshot service is used for displaying the needed section of the map. Using the snapshot service the optimized routes will be loaded much faster as only the certain parts of the map will be refreshed with the selected path.

V. IMPLEMENTATION

***A:** Google Direction API:*

Google Maps is a web-based mapping service provided by Google which provides a highly responsive visual interface built using AJAX technologies. This service has detailed street and aerial imagery data and an open API allowing customization of the map output including the ability to add application specific data on the map and allow developers to integrate Google Maps into their websites with their own data points and it is a free service

1. Overlays – representing POI (point of interest) on map or lines denoting areas, routes, or other information about the location being displayed.

```
{
        featureType: "road.highway",
         elementType: "geometry", "
         stylers: [ { "hue": "#44ff00" } ]
 }
```

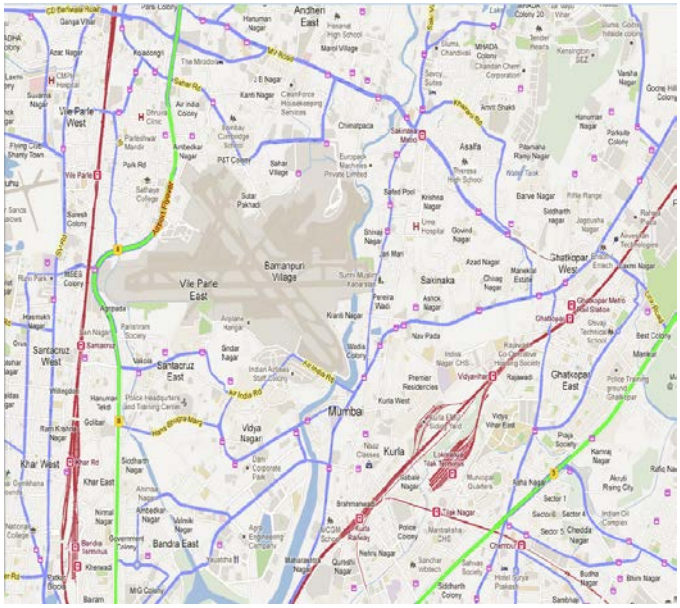
2. Events – are triggered by some action, such as the user click or mouse over on POI or some other action associated with GEvent. Info Windows are commonly called with GEvent and click action.

```
google.maps.event.addListener(marker, 'click', function()
{
   map.setZoom(8);
   map.setCenter(marker.getPosition());
}
```

3. Controls – basic interface controls that enable user to zoom in and out the map, move the map effectively, and ability to change between map types.

var mapOptions =
{

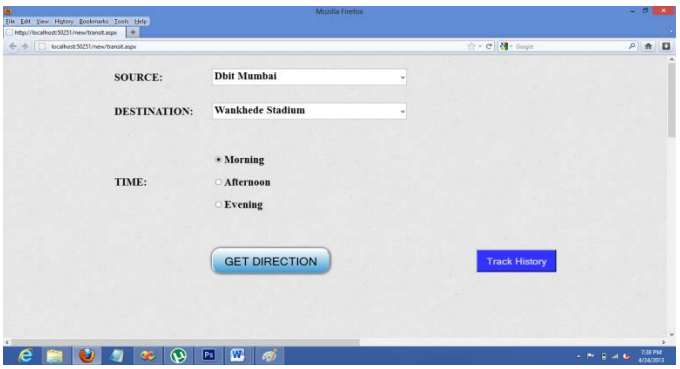
zoom: 4,
center: new google.maps.LatLng(-25.363882, 131.044922),
mapTypeId: google.maps.MapTypeId.ROADMAP
}



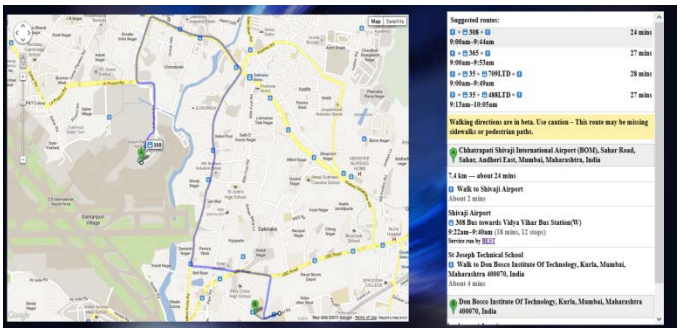
Google Maps (Layered) (figure 2)

4. Services – extends Google Maps API with adding new functionalities and features that are often available on *maps.google.com* first.

4.1 Transit Service:



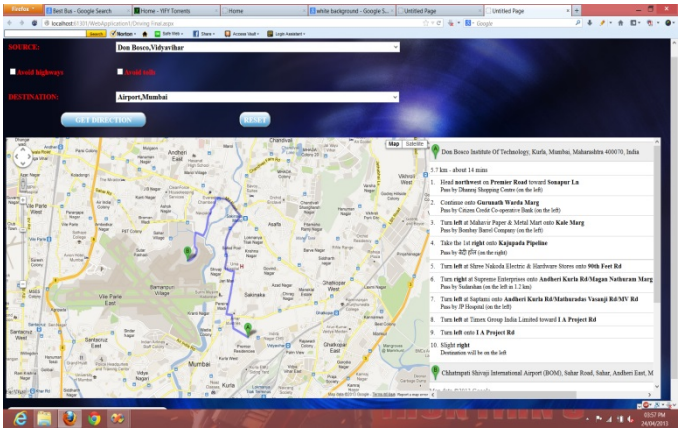
Transit (Select Options) (figure 3)



Output (figure 4)

provideRouteAlternatives: true,
unitSystem: google.maps.UnitSystem.METRIC,
travelMode: google.maps.DirectionsTravelMode.TRANSIT,

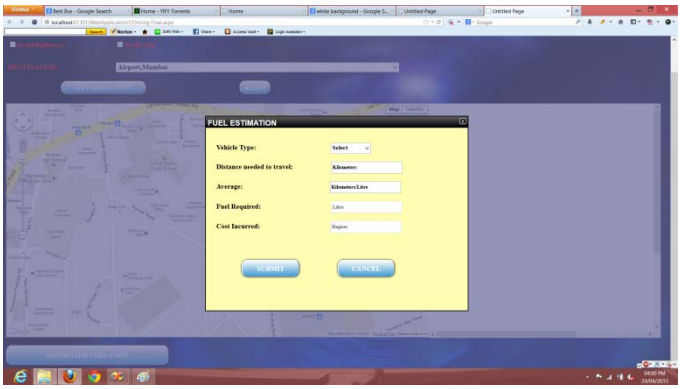
4.2 Driving:



Google Maps (Select Options and Output) (figure 5)

provideRouteAlternatives: true,
unitSystem: google.maps.UnitSystem.METRIC,
travelMode: google.maps.DirectionsTravelMode.DRIVING,

4.2.1 Fuel Estimation:



Fuel Cost and Estimation (Driving) (figure 6)

<cc1:ModalPopupExtender ID="ModalPopupExtender1" BackgroundCssClass="ModalPopupBG" runat="server" CancelControlID="Button4" TargetControlID="Button5"

PopupControlID="Panel1" Drag="true" PopupDragHandleControlID="PopupHeader" OnCancelScript="clear()"> </cc1:ModalPopupExtender>

B: SQL:

An SQL join clause combines records from two or more tables in a database. It creates a set that can be saved as a table or used as it is.

Equi-join:

An equi-join is a specific type of comparator-based join that uses only equality comparisons in the join-predicate. Using other comparison operators (such as <) disqualifies a join as an equi-join. The query shown above has already provided an example of an equi-join:

Cost Estimation-Transit (figure 7)

The grid view shown here is the Cost associated with the optimal travel routes provided by Google API. The four alternative routes are obtained by the following query:

SqlCommand cmd2 = new SqlCommand

("select t.Route_info

[ROUTE INFORMATION],t.cost_of_bus,t.cost_of_train,

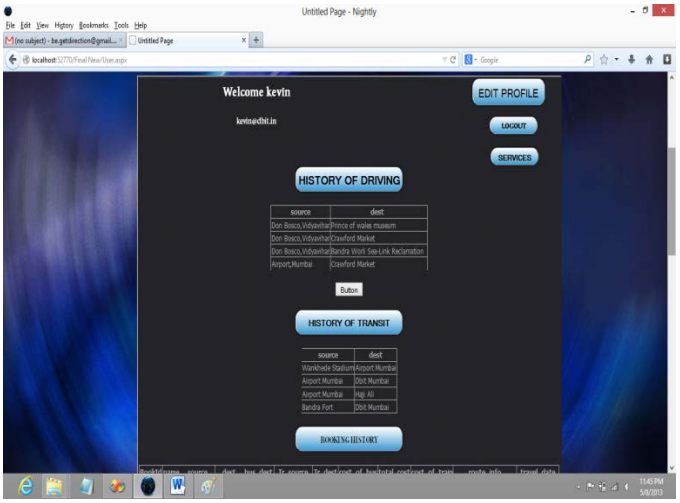
t.Total_cost [TOTAL COST]

from travel t

where t.schedule=" + str + " and t.sid=(select sid from source p where p.source='" + src + "' and p.dest='" + dest + "')

All the routes are given a unique sid. With the help of source and destination selected from their respective DropDownList only the alternative paths will be retrieved and displayed on the gridview as shown above.

C: Session Tracking:



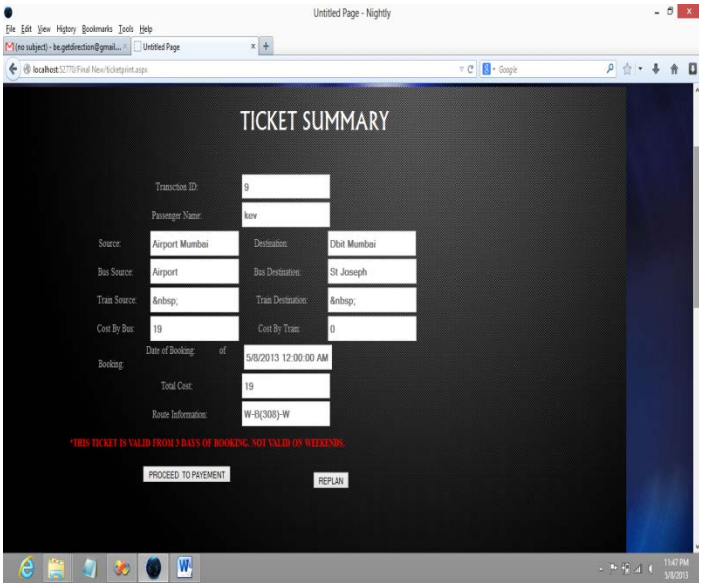
User Page (figure 8)

<asp:SqlDataSource ID="SqlDataSource3" runat="server" ConnectionString="<%$ConnectionStrings:ConnectionString2 %>"SelectCommand= ""SELECT [BookId], [name], [source], [dest], [bus_dest], [Tr_source], [Tr_dest], [cost_of_bus], [total_cost], [cost_of_train], [route_info], [travel_date] FROM [booking] WHERE ([Tranid] = @Tranid)"]"></asp:SqlDataSource>

Admin Page (Transaction History) (figure 9)

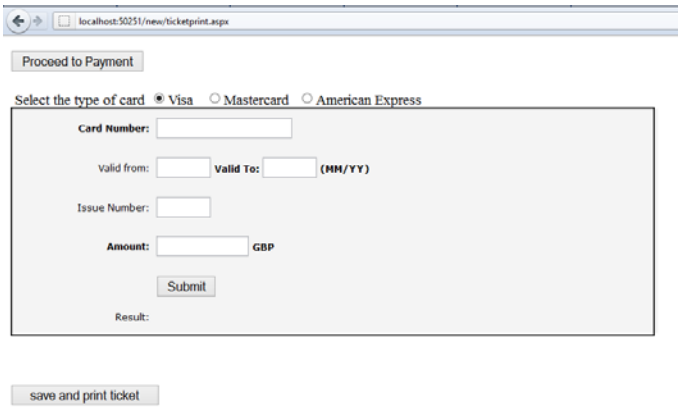
<asp:SqlDataSource ID="SqlDataSource3" runat="server" ConnectionString="<%$ConnectionStrings:ConnectionString2 %>"SelectCommand= "SELECT [name], [source], [dest], [bus_dest], [Tr_source], [Tr_dest], [cost_of_bus], [cost_of_train], [Tranid], [total_cost], [route_info], [travel_date] FROM [booking]"></asp:SqlDataSource>

D: Payment and Ticketing:



Ticket Page (figure 10)

```
DateTime today = DateTime.Today;
TextBox8.Text = Session["routeinfo"].ToString();
TextBox5.Text = Session["totalcost"].ToString();
TextBox7.Text = Session["source"].ToString();
TextBox2.Text = Session["dest"].ToString();
TextBox9.Text = Session["name"].ToString();
TextBox6.Text = Session["userid"].ToString();
TextBox3.Text = Session["costofbus"].ToString();
TextBox4.Text = Session["costoftrain"].ToString();
TextBox14.Text = today.ToString();
TextBox.Text = Session["bussource"].ToString();
TextBox11.Text = Session["busdest"].ToString();
TextBox12.Text = Session["TrainSource"].ToString();
TextBox13.Text = Session["TrainDest"].ToString();
Amount.Text = Session["totalcost"].ToString();
```



Payment (Card Authentication) (figure 11)

```
<etier:CreditCardValidator
 Id="MyValidator"
 ControlToValidate="CardNumber"
 ErrorMessage="Please enter a valid credit card number"
 Display="none"
 RunAt="server"
 EnableClientScript="False"
 ValidateCardType="True"
 AcceptedCardTypes="Amex" />
```

On clicking the Print Ticket, the user will be directed to a page where he can print and save the ticket. The user is asked for his credit card details and transaction is validated using a credit card validator. When the user clicks the Print button all the information required in the ticket is stored is retrieved using a concept called session in the following way:

```
Session["source"]= DropDownList1.SelectedItem.Text;
Session["dest"]= DropDownList2.SelectedItem.Text;
Session["routeinfo"]= GridView1.SelectedRow.Cells[1].Text;
Session["totalcost"]= GridView1.SelectedRow.Cells[4].Text;
Session["costofbus"]= GridView1.SelectedRow.Cells[2].Text;
Session["costoftrain"]=GridView1.SelectedRow.Cells[3].Text;
```

VI. HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Specifications
1. The system should be embedded in the PC/Laptop.
2. 40 GB hard disk and 256 MB RAM
3. Processor- Intel Pentium

Software Specifications
1. Language : - C#, JavaScript, HTML
2. Backend : - Microsoft SQL Server, Visual studio 2008.
3. Adobe Photoshop CS5
4. Operating System : - Windows XP or above
5. Internet Browser : - Mozilla Firefox, Internet explorer, Google Chrome.

VII. TESTING AND DEBUGGING

This web application passed the following tests:

1. Cross-browser compatibility testing; addresses browser compatibility and operating system compatibility.

2. Functionality testing; covers basic web site functionality as links, forms for information submitting, database connections, cookies.

3. Usability testing; covers user interface and navigation.

4. Interface testing; covers interaction between application, web server and database server.

5. Database Testing: Validity and integrity testing, performance related to database. Testing of procedures triggers and functions.

6. Map testing: All the location that the website will be using is working as per expected.

7. Performance and stress testing; application performance evaluation during high load and user activity, testing application response on different internet connection speeds

VIII. CONCLUSION

Thus we have created a website which allows a user to manage his expenses and time efficiently and reach a desired destination and in case of transit mode, also book the tickets. The realized application uses the benefits of Google Maps API service, also adding the feature of layering the maps with different color for showing different objects and is developed using ASP.NET. Currently our project provides travel information only for limited number destinations in Mumbai. As a part of future work the system can be enhanced by adding more destinations to the database, providing travel information and booking tickets for weekends. Also contact details for hiring cool cabs.

IX. REFERENCES

[1] Francis Rousseaux, Kevin Lhoste, "Rapid Software Prototyping Using Ajax and Google Map API," *achi, pp.317-323, 2009 Second International Conferences on Advances in Computer-Human Interactions, 2009.* [Dec 2012].

[2] Martin C. Brown, “Hacking Google Maps and Google Earth”, *ISBN: 978-0-471-79009-9, 2006.* [Aug 2012]

[3] Google Maps API Documentation, http://code.google.com/apis/maps/documentation/. [Jan 2013]

[4] Konarski, Michał, Zabierowski, Wojciech, “Using Google Maps API along with technology .NET”, Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), 2010 International Conference. [Jan2013]

[5] Pejić, Aleksandar, “An Expert System for tourists using Google Maps API”, Intelligent Systems and Informatics, 2009. SISY '09. 7th International Symposium. [Jan 2013]

[6] https://developers.google.com/ [Dec 2013]

[7] http://www.codeproject.com/Articles/291499/Google-Maps-API-V3-for-ASP-NET [Jan 2013]

[8] http://www.daniweb.com/web-development/aspnet [March 2013]

[9] www.bestundertaking.com/ [March 2013]

[10] http://www.lynda.com/ [Oct 2012]