DEVELOPMENT OF DYNAMIC ROUTING SYSTEM FOR TOURISM BASED ON WEB MAP APPLICATION USING FOSS4G AND PGROUTING

TENZIN WANGCHUK

A Thesis Submitted to the Graduate School of Naresuan University
in Partial Fulfillment of the Requirements
for the Master of Science in (Geographic Information Science)
2019

DEVELOPMENT OF DYNAMIC ROUTING SYSTEM FOR TOURISM BASED ON WEB MAP APPLICATION USING FOSS4G AND PGROUTING

TENZIN WANGCHUK

A Thesis Submitted to the Graduate School of Naresuan University
in Partial Fulfillment of the Requirements
for the Master of Science in (Geographic Information Science)
2019

Thesis entitled "Development of dynamic routing system for tourism based on Web Map Application using FOSS4G and pgRouting"

By TENZIN WANGCHUK

has been approved by the Graduate School as partial fulfillment of the requirements for the Master of Science in Geographic Information Science of Naresuan University

Oral Defense Committee

.... Chair
(Assistant Professor Pipat Reungsang, Ph.D)

.... Advisor
(Assistant Professor Sittichai Choosumrong, Ph.D.)

.... Internal Examiner
(Assistant Professor Kampanart Piyathamrongchai, Ph.D.)

Approved

....
(Professor Paisarn Muneesawang, Ph.D.)

for Dean of the Graduate School

Title DEVELOPMENT OF DYNAMIC ROUTING SYSTEM FOR TOURISM BASED ON WEB MAP APPLICATION USING FOSS4G AND PGROUTING

Author TENZIN WANGCHUK

Advisor Assistant Professor Sittichai Choosumrong, Ph.D.

Academic Paper Thesis M.S. in Geographic Information Science, Naresuan University, 2019

Keywords Dynamic Routing, Tourism Based, Web Map, FOSS4G, pgRouting

ABSTRACT

Most map services do not support adding points or area incapacitating the routing service of a dynamic feature and providing real time scenarios. They often do not even provide important geospatial routing functionality like overlay and interpolation. This paper attempts to develop a Web GIS map service using a Free and Open Source for Geospatial (FOSS4G) and pgRouting to provide a dynamic routing system for improved response time, accomplish an effective routing and to show the Point of Interest (POI) along the shortest route as a solution to the ever increasing traffic in the capital city of Bhutan, Thimphu.

Towards building a smart city, use of such technology will enhance service delivery and information sharing to the tourist who are visiting Thimphu and even the residents. The tourist and residents can also use it to locate the shortest route towards their destination. It will be helpful to find essential services available in the town like medical and police services. The framework and method of the study can be used for disaster prevention and related information. The study could be replicated in the other major cities; Phuntsholing, Samdrup Jongkhar and Gelephu.

ACKNOWLEDGEMENTS

Firstly I would like to thank my supervisor, Assistant Professor Dr.Sittichai Choosumrong, Department of Natural resources and environment ,Naresuan University, who supervised this study and guided me throughout the research work. It is his untiring assistance and expertise that showed me the right direction, which besides the academic achievement will also help in my personal and professional career. His guidance, suggestions and new ideas enhanced my skills on how to work with Open Source Software.

I would also like to thank Assistant Professor Dr.Kampanart Piyathamrongchai for his valuable advice.

I further extend my thanks to Associate Professor Mr. Pathana Rachavong, Assistant Professor Capt.Dr. Chaiwiwat Vansarochana, Assistant Professor Dr.Natapon Mahavik, and Lecturer Ms.Tanyaluck Chansombat for their advice and supports .

Finally, my sincere thanks to Department of Agriculture Natural Resources and Environment, Faculty of Agriculture Natural Resources and Environment, Naresuan University and teachers for supporting me throughout my two years stay. I also would like to express my sincere gratitude to Thailand International Cooperation Agency (TICA), Royal Government of Bhutan and National Statistics Bureau for my scholarship to study in Thailand.

Lastly my heartiest gratitude to my loving family and friends for their continues support, care, love and encouragement that made me possible to achieve my goal. It was not possible to come up with this report and complete my studies without their moral and endless support.

TENZIN WANGCHUK

TABLE OF CONTENTS

Page

List of tables

Page

List of figures

Page

CHAPTER I

INTRODUCTION

1.1 Motivation and Background

The need for application in Geoinformatics gained importance for easing the day to day activities. Development of Geographic Information System (GIS) and their web-based application have been increasing because the use of GIS applications makes our life much easier. Among many GIS applications, web route mapping is one of the web base applications used to access the map service via a web browser and smartphone. There are so many common routing services like google direction API, Grass hopper direction, Mapbox direction and open route services but current routing systems in GIS software mostly provide routes that allow users to navigate between source and destination points only, but in real life there can be natural calamities like flood, earthquake and road accidents which will result in blocking the road. So in this kind of scenario the web application resulting routes do not take into account recent changes in the network data and even with this API of Google maps do not provide features like overlay, interpolation etc (Pritee & Garg, 2017). The implication of this problem can be loss of time and fuel, and especially in case of emergencies the real road network conditions should be taken into account for the computation, by representing not only the relationships between the transportation elements, but also the real-time and dynamic traffic restrictions in the road network (Choosumrong Sittichai, 2010).

The topic of the study is to Development of dynamic routing system for tourism based on Web Map Application using FOSS4G and pgRouting. In this study, a method of finding the shortest route in a real time situation of the road and to find the Point of Interest (POI) along the shortest path was developed.

The main aim of the study is to build a dynamic routing web application by using pgRouting algorithm. The system needs to keep the updated road situation to find the shortest path. For example, if there is a road accident, road block or road maintance it should be updated. The system developed will be helpful for traffic management, disaster and tourism traffic management (Choosumrong Sittichai, 2019). The study will be using the standard of FOSS4G like MapServer, PHP, PostgreSQL, PostGIS, Web Map Service, and Web Feature Services.

There is an urgent need of innovative initiatives for developing smart cities. The strength of Information Technology needs to be capitalized to ensure accessible and efficient delivery of public services. Bhutan is the most rapidly urbanizing country in the South East Asia. Therefore, to keep abreast of the fast-moving developmental activities, it is important to have a very well thought planning process (Longo & Roscia, 2014).

Unprecedented growth of Thimphu City coupled with inadequate urban infrastructure has resulted in a shortfall of basic services to its residents. Infrastructure growth has been increasing but access to water, sanitation, solid waste management, and urban transport are often inadequate.

Since the Tenth Five-Year Plan invested in the infrastructure and management in two major cities; Thimphu and Phuentsholing and other larger urban centers to ensure sustainable urban management. Thimphu Municipality is Bhutan's capital city, with a population of 114,551consituting almost 42% of urban population. The Thimphu city development plan identifies congestion and poor urban mobility as a key issue that restricts the city's growth. Therefore, it is crucial to bank upon technology to make the transport system efficient.

The Thimphu Municipality strives to build a very smart city as per the Thimphu Structure Plan. The fiscal year 2019-2020 annual performance target prioritized smart city initiatives as one of the targets. Thus, the development of dynamic routing web application will contribute in a little way in Thimphu Municipalities aim of building a smart city (Hendawi et al., 2017).

1.2 Research Objectives

The Main objectives of the study are as follows:

1. Design and develop a Web GIS using Free and Open Source Software to find the shortest route.
2. To implement Location Based Service (LBS) system to serve the Point of Interests (POI) along the shortest route for the tourist people.
3. To develop a dynamic routing system for real-time navigation and minimizing the risk of driving through a temporarily blocked road.

1.3 Problem Statements

The number of vehicles in Thimphu city have been increasing over the years, a total of 41,562 vehicles were registered in June 2016 and by June 2019 it has increased to 53,999. An absolute increase of 12,437 over the duration of 4 years or an annual increase of about 3,000 vehicles. The increase may not seem as an issue but for a total road length of 250 KM with very less intentions of increasing calls for actions to make routing easier in the future is a cause of concern. The main roads within the core town area remains crowded and congested during the rush hours of 8:30 am – 9:30 am and 5:00 pm – 5:30 pm. The situation gets worse when national events are being organized (Road Safty and Transport Authority, June 2019).

Besides, there is an ever-increasing number of tourists visiting Bhutan. The total tourist arrival in 2018 was 274,097 with a growth of around 8 percent over 2017. Of the total, 202,290 were regional tourists and 71,807 were international tourists inclusive of leisure, official, business and others. The impact of increasing tourist arrival is already visible with many regional vehicles plying on the highways and major cities. This adds to the already existing traffic problem especially in Thimphu. Moreover, with the increasing number of regional tourists since 2013, there is also a need for geo-info system to help the tourist find the way towards popular destinations. This is crucial because majority of the regional tourist do not visit through registered agents and thus there is no guide accompanying them (Bhutan, 2018).

With the increasing number of vehicles, the area is more prone to accident and traffic congestion, if there is a temporarily block there should be a system where the information needs to be updated on the data base. By updating the database, we can control the loss of fuel, time and especially during emergencies. This problem can be solved by Creating a dynamic routing system based on Web GIS that will improve response time in case of emergencies and for normal users as well (S. Singh et al., 2015).

CHAPTER II

REVIEW OF RELATED LITERATURE

2.1 Network Analysis

A network can be considered to be a pure network if their topology and connectivity are being considered. If a network is characterized by its topology and flow characteristics (such as capacity, path choice and link cost functions) it is referred to as a flow network. A transportation network is a flow network representing the movement of people, vehicles or goods (Fischer, 2004).

Network analysis is one of the major important function of GIS. The function includes shortest path analysis, resource allocation and etc. From the functions shortest path analysis is one of the basic and most important function in GIS (Xie Dexiang, 2012).

There are two kinds of routing

- Static Routing.
- Dynamic Routing.

2.2 pgRouting

pgRouting is an open source library that provides different tools to search the shortest path. pgRouting is an extendable open-source library that provides a variety of tools for shortest path search as extension of PostgreSQL and PostGIS. In the beginning it was called as pgDijkstra because it implements only the shortest path search but later other functions were also added (Project).

The pgRouting can be extended with PostGIS/PostgresSQL geospatial database to provide geospatial routing functions. Using of this routing functions gives the following advantages: -

- Multiple clients through JDBC, ODBC or directly using PostgreSQL API can be easily accessible.
- Without pre-calculation, data can be changed instantaneously through the routing engine by using OSS like QGIS.

- The cost parameter can be directly calculated by using SQL and its value can come from multiple fields or table.

Although pgRouting in the beginning was designed to find the shortest path queries and small networks, but now additional functions are easily integrated. The library contains the following functions: -

- All Pairs Shortest Path, Johnson’s Algorithm
- All Pairs Shortest Path, Floyd-Warshall Algorithm
- Shortest Path A*
- Bi-directional Dijkstra Shortest Path
- Bi-directional A* Shortest Path
- Shortest Path Dijkstra
- Driving Distance
- K-Shortest Path, Multiple Alternative Paths
- K-Dijkstra, One to Many Shortest Path
- Traveling Sales Person
- Turn Restriction Shortest Path (TRSP)

In this study Shortest Path Dijkstra is chosen to modify the function to get the best routing result.

2.2.1 Shortest Path Dijkstra Algorithm.

Dijkstra algorithm was the first algorithm implemented in pgRouting and its also called as a Greedy algorithm (Xie Dexiang, 2012). Dijkstra's algorithm starts from a source node, and in each iteration adds another vertex to the shortest-path spanning tree. Dijkstra care only the sum of their weights (finds the path with lowest cost) between that vertex and every other vertex.

2.2.2 Calculation for the Shortest Path Dijkstra Algorithm.

The Shortest Path Dijkstra distance is calculated as follows:

Step I: Mark your selected initial node with a current distance of 0 and the rest with infinity.

Step II: Set the non-visited node with the smallest current distance as the current node C

Step III: For each neighbor N of your current node C add the current distance of C with the weight of the edge connecting C-N If it's smaller than the current distance of N set it as the new current distance of N

Step IV: Mark the current node C visited

Step V: If there are non-visited nodes, go to step 2

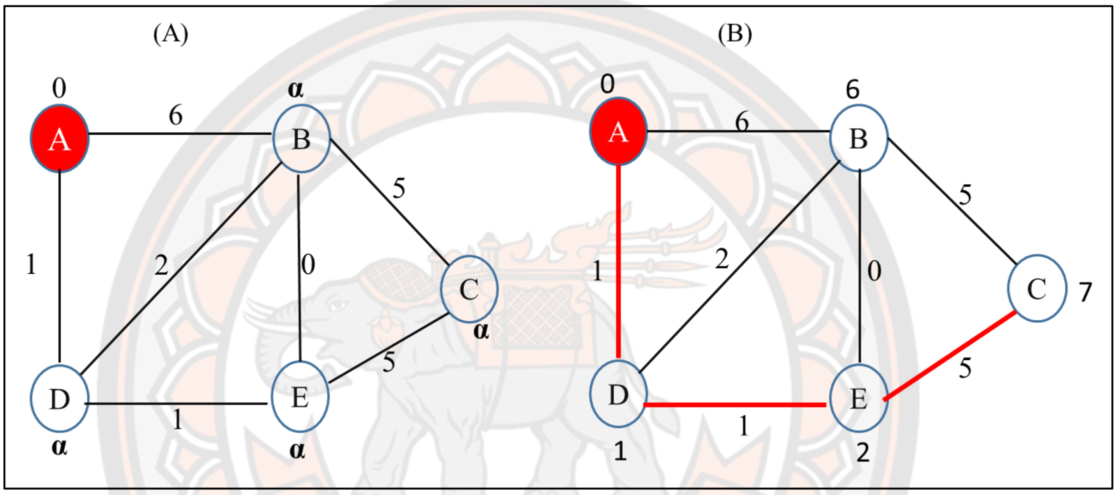


Figure 1 Basic Shortest Path Searches A) Initial cost of the network B) Cost values from node A to C

From the graph in figure 1 we can get the shortest path from starting node (Node A) to each node. The values are A to B = 6, D=1, E=2, and C= 7.

2.3 Examples of different API and routing algorithm used in Routing system.

There are many different API and routing functions used for routing system. The following are some examples being explained: -

2.3.1 Travelling Salesman Problem

Travelling Salesman Problem (TPS) is one of the library in pgRouting which is used to find the shortest path. The main aim is to find the shortest or best route when a sales man starts a journey from a home city to another given cities and to come back to the home city by just visiting once. In TPS there has no best optimal algorithm for all the cases, efforts are still being used to find the optimal solution for specific cases (Shaw & Gurram, 2015): -

- In the year 1954 TPS path have been used for 49 cities in USA was solved.
- In 1977, 120 cities in West Germany.
- With the development in information Technology, TPS path to 13509 cities in USA.
- In 2004 the optimal tour of 24978 cities in Sweden.

TPS is one of the widely used algorithm which is used to solve the problem in mathematics because of its widely accepted in social life, for example: -

Application in Management: -

- Bus route in the city.
- Collection of coins from vending machines.
- Tourism route for the Tourist

2.3.2 Google Direction API

Google Direction API is a services which calculates direction between two locations. We can search for direction on a different mode for transportation like driving, cycling and walk. Direction API through an HTTP interface request to construct the URL string by either using text string or the coordinates to identify the locations. Apart from Google Direction API there are so many direction API like Graphhopper direction, Mapbox direction and Openrouteservices which serves the same purpose (Haitao et al., 2019).

Whereas for my study I have used Shortest Path Dijkstra function from pgRouting algorithm because we can modify the code and we can test on the free and open source software.

2.3.3 OpenRouteService

OpenRouteService is much more than a routing service, it uses a wide range of services based on Open Street Map data which can be used in all different kinds of applications. The following services used in the framework of OpenRouteservice (Feng et al., 2011):

- Directions Service determines travel routes and navigation information according to different criteria. This has been based on as follows:
    - Cars: fastest, shortest, recommended
    - several options to avoid tools, tunnels, etc.
    - multiple heavy vehicles profiles (Delivery, Forestry, Bus) with many customizable options
    - bicycles (MTB, race bike, safest route and more)
    - pedestrian (normal and hiking)
    - wheelchair routing
- POIs Service is a service that provides access to an online directory to find the location of a specific or nearest place, products or services.
- Isochrones Service calculates a polygon representing the area that is reachable within a certain time distance based on a street network around a given location.
- Geocode Service provides a Geocoder/Reverse Geocoder; the Geocoder transforms a description of a location, such as a place name, street address or postal code, into a normalized description of the location with a Point geometry.
- Polygons can be digitalized on the map which will be avoided for subsequent routing.
- Upload and Download of GPS Tracks in different formats.
- For the Pedestrian and Bicycle Profiles type of the surface, type of ways and suitability for the selected profile as well as a height profile can be shown.

2.4 GIS and the Web.

The new generation of Information Technology has come up with more advance methods of hosting a different kind of web site. In the field of Geographic, it has come up with a new method of hosting a web called as a Web GIS. It was started in the year 1993 in an era called as web 2.0 era.

Web GIS is a process of designing, implementing, generating and delivering maps on the World Wide Web. GIS integrates and relates data with spatial components and support users to view the data in the form of maps which helps in making the decision through visualization. Web GIS is a GIS application which is developed and made it available through a common web browser. Together with the World Wide Web, GIS can be used to further develop and allow people to access to GIS functionality so that it can enhanced the people participation in planning.

Web GIS has four major system components, which includes client for sending request to the web browser, web server with the application to respond to the request, map server and data server. These four components are used to develop the Web GIS for the study. Web Based GIS is one of the newest features in Geographic Information System for creating a web base map. There are so many new technologies and software use to create a Web GIS. We have the commercial, free and open source software (FOSS) and public software. People tend to use more FOSS software because of involvement of large number of people and can be used free of charge.

2.4 Web GIS Architecture

With the development of Information Technology, Web GIS has become very popular now days. Web GIS is a GIS that uses a web technology. It has a Web browser as a client, for sending a request and a Web server for responding the request. Usually all the web application has a we2.b server, but in Web GIS since it has a huge amount of data it needs an extra server called as a Map Server. Map server handles the data.

Generally, the architecture of web GIS has three layers namely the interface layer, application layer and the data base layer. The client accesses the system through the interface layer which takes the input and then shows the output on the system. Application layer shows the map visualization functions like panning, zooming etc. and even other functions like selection, editing and querying etc. Database layer handles the data and how it is being presented and distributed (Bendib et al., 2016).

2.6 PHP

There are so many web being created by using different languages like Perl, UNIX shell program, C, C++, Cold fusion, inter-base, java script and java but most of them has their own advantages and disadvantages. But now a day most of the web are built by using a very common language called as a PHP3.

PHP3 was developed in the year 1994, the first public version was available in the year 1995.It was called as a Personal Home Page Tools (PHP Tools). It was however upgraded and named it as PHP/FI version 2 in mid of 1995.Another package Rasmus had written that interpreted HTML form data and was called as Personal Home page tools script with FI. Then in 1997 PHP3 changed from Rasumus personal project to Open Source Development. Now it is an Open Source (PHP; Prokofyeva & Boltunova, 2017).

PHP3 is an independent of the operating system, the web server, HTML and the data base. It has got some of the functions from other languages like C, Java, Perl and UNIX shell. PHP3 can used for following: -

- Create different dynamic parts for Web GIS application
- Used to run spatial SQL queries on the database level and can add the result to the graphical interface of the web.
- Run attribute queries and results as graphics on the web application.
- Add security to the Web GIS application

2.7 Database Management System

Database management system (DBMS) is a system software used for creating and managing the database. DBMS is a program which makes the user to create, update and retrieve the data. It also works as an interference between data base and the application program ensuring that the data is organized and is easy to access. The advantages of using DBMS, the data is being protected and maintained. While sharing the data from DBMS instead of creating a new iteration of the same data stored in new files for every new application. It also provides a central store of data that can be accessed by multiple users in a controlled mannered. The most commonly use Database Management system used now a days are Relational Database Management System (RDBMS). Structured Query Language (SQL) which is used for managing and querying the database are mostly used in RDBM (Chunithipaisan, 2010).

2.8 GeoServer

GeoServer is an open-source server written in Java that allows users to share, process and edit geospatial data. By using open standards set by the Open Geospatial Consortium (OGC), GeoServer allows for great flexibility in map creation and data sharing. It is a central part of a system which allow the system and displays the client request in various format through WMS, WFS and GeoJSON specification.

GeoServer allows us to display our spatial information to the world. Implementing the Web Map Service (WMS) standard, GeoServer can create maps in a variety of output formats like OpenLayers, Leaflet etc. Which is integrated into GeoServer, making map generation quick and easy. GeoServer is built on GeoTools, an open source Java GIS toolkit.

GeoServer conforms to the Web Feature Service (WFS) standard, and Web Coverage Service (WCS) standard which permits the sharing and editing of the data that is used to generate the maps. GeoServer also uses the Web Map Tile Service standard to split your published maps into tiles for ease of use by web mapping and mobile applications.

2.9 PostGIS/PostgresSQL

PostGIS is a powerful open-source tool that allows to develop robust spatial databases. PostGIS can be considered an extension of database management system (DBMS) of PostgreSQL that can manage spatial data. PostGIS enables us to store geographic objects as part of our data tables (Lizardo & Davis Jr, 2017).
PostGIS is one of the most reliable open-source DBMS. The advantages over other alternatives are listed below: -

- Its ability to build indexes in any kind of data, having a generic index structure.
- It has many useful spatial functions to search, analyze, convert and manage spatial data.
- It has both vector and raster data support.

  - Vector data represents phenomena in the world in terms of points, lines and polygons.
  - Raster data represents the world in terms of cells of predefined, grid-shaped tessellations.
- It's based on open standards as defined by the Open Geospatial Consortium.
- It's supported by other well-proven open-source projects, such as Proj4, Geometry Engine - Open Source (GEOS), and Geospatial Data Abstraction Library (GDAL).
- It is compatible with almost all major open- and closed-source GIS Software.

2.10 Leaflet and Openlayer

Leaflet is one of the leading open-source JavaScript library used for mobile-friendly interactive maps. It has all the mapping features most developers need.

Leaflet is designed with features like simplicity, performance and usability. It works effectively with all major desktop and mobile plate forms. It has lot of plugins that can be easily used and has a well-documented API. The source codes are simple and easily readable (Han, 2018).

OpenLayers is an Open Source JavaScript, released under the 2-clause BSD License (also known as the FreeBSD) use to make the dynamic map on any web page. It can display map tiles, vector data and markers loaded from any source. OpenLayers has been developed to be used with geographic information of all kinds. This software supports in different variety of data source under OGC standards, such as WMS or WFS (Chandniha et al., 2017).

2.11 Review

The research conducted by (Choosumrong Sittichai, 2019 #9) on development of optimal routing service for emergency scenarios using pgRouting and FOSS4G to facilitate pervasive dynamic routing services on road network data, demonstrated an algorithm for dynamic routing which overcame several limitations such as ERDP functionality was developed by integrating Dijkstra shortest-path routing algorithm and AHP. New and improved weighted travel-time algorithm was implemented by extending the functionality of the Open Source pgRouting library. An algorithm had developed as a Web application using OpenStreetMap data and FOSS4G tools such as PostgreSQL, PostGIS, MapServer and Openlayers. The positioning ERDP as Web application enabled platform independence and easy dissemination of routing resulted the based on up-to-date road network data. (Choosumrong et al., 2012) highlighted that using an improved algorithm had implemented by extending PgRouting to acquire the road situations at the destination and offer an emergency route decision planning by computing the consumption of travel time of route on the basis of location and situations of accidents. Application of pgRouting in road and highway level delivered a rapid speed, quick execution and appropriate operation for finding the shortest path and extensive use in path optimization. On the basis of this absorption, it provided a benefit of traveling salesman problem solution and highway grade analysis (Zhang & He, 2012) Similar study done by (Singh et al., 2015)using pgRouting on dynamic shortest route finder for emergency management showed that the system presented can be used to obtain alternate paths by identifying the affected regions as some form of obstruction for emergency management. This can be further extended by calculating an ideal shortest path, taking into account of the dynamic cost factors encompassing slope, width, and road type.

The study on smart personalized routing for smart cities using Prego and SP-TAG by (Hendawi et al., 2017)found that PreGo was considerably more effective than the SP-TAG Best-Start from the CPU time because PreGo combined all attributes in one run. For the CPU time, it was observed that both algorithms had been increased with the trend. However, PreGo was increasing slightly with smaller values than the SP-TAG which augments drastically with much larger values. They also found that with regard to the memory consumption, both of them were marginally increasing but the cost of PreGo was increased with PreGo which attributed to the joined set of attributes in one run. This was in contrast with a single attribute at a time in SP-TAG start. Authors confirmed PreGo had provision in optimizing start times, and incorporating user’s weighted preferences which were key features missing from several research and commercial frameworks. The PreGo within algorithm performed 100 times faster than the competitive technique.

According to Agrawal and Gupta (2014) integration of city GIS data with Google Map API and google earth API for a web based 3D geospatial application showed that Google Earth Map was exhibiting 3D spatial objects and Google Map was revealing attribute data of the building using info windows. Google provided satellite images, road maps, terrestrial maps, and other geocoding options, because of this, in web based geographical information systems, the Google Earth and Google Map was a good source and option for users. This web based GIS application showed how Google Map integrated with Google Earth API can be used for a comprehensive web based 3D city models. By storing the corresponding attribute data in XML database, the detailed information about a location on the 3D map was exhibited which delivered users a more realistic experience and information for user needs. Integrating with the Google Map allowed road directions, street names and all other location information to be obtained together with the 3D buildings. This application will increase productivity and information sharing for private users, local businesses and the general public by making it easy to view, analyze, and make maps with authoritative local geographic data.

As per Xie Dexiang (2012) research on Dijkstra algorithm to generate the shortest path cost from source node to all other nodes. They had used to improve the efficiency of the Dijkstra algorithm and found to improve the storage and used of efficiency searching method. But overall, the improved algorithm was found to be better than the performance of the original algorithm. The similar study done by on (Kumari Pritee, 2017) Dijkstra algorithm application: shortest distance between buildings unfolded the way that anyone can use Dijkstra algorithm to obtain the shortest path to reach the destination of any regions in which they live using the algorithm process method. (Pritee & Garg, 2017)also carried out similar research using multipath Dijkstra’s algorithm for identification of optimum shortest path. They found Dijkstra’s algorithm very effective and appropriate operation with vigorously updating cost. It benefited for traffic conditions, studying road conditions and planning journey. It also enlarged and discovered various routing functions, used in PostgreSQL on the basis of three factors which include shortest, fastest, traffic free, alternative route in obstacles and natural route. The application was applied on web platforms like GeoServer and Geoexplorer because of the high penetration of both technologies.

The research carried out on WebGIS with google maps by (Lange & Plass, 2008) found that the new system WebGIS offered more functional that generated new geographic datasets from current datasets. The WebGIS had advantage features such as they were easy to handle, free access for everyone, especially easy and free access to geodata (Google Maps), easy capturing and editing, analyzing new geodata. Further work can be implemented on advanced spatial analysis functions.

CHAPTER III

RESEARCH METHODOLOGY

3.1 Study Area

The area for my study is Thimphu Town/ Municipality, Thimphu District, Bhutan. it's also known as Thimphu Thromde which is located at 27°28′00″N 89°38′30″E Coordinates: 27°28′00″N 89°38′30″E and is spread over an altitudinal range between 2,248 meters (7,375 ft) and 2,648 meters (8,688 ft). Thimphu town covers an area of 26 Sq.km with a population of 114551 as per the Population and Housing Census of Bhutan (PHCB) 2017.

Thimphu as the capital city of Bhutan have the highest number of population. Consequently, the numbers of vehicles imported in the city has being drastically increasing over the years. The increasing number of vehicles makes the city very venerable to road accidents and traffic congestions.

Therefore, the dynamic routing system proposed in the study can improve the transportation system in the city which will be helpful for both the local residents and more so for the tourists.

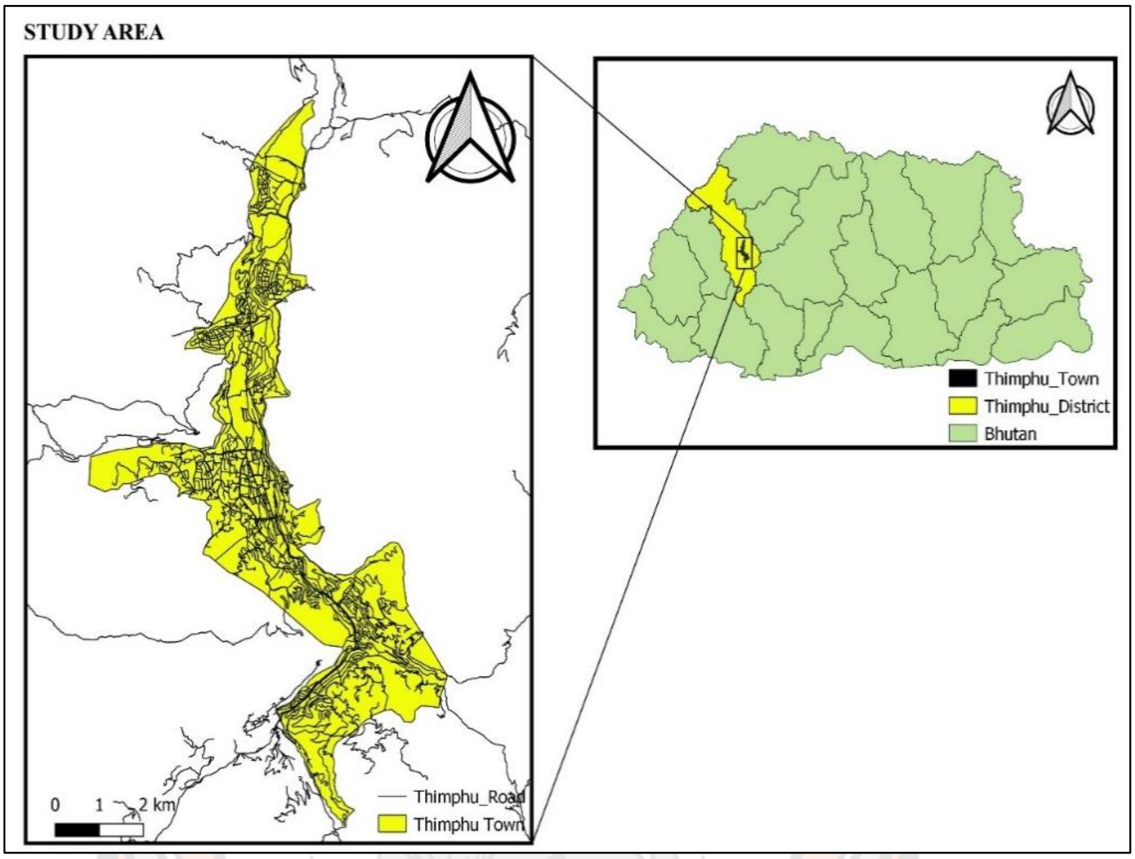


Figure 2 Location of Thimphu District with Municipal, Bhutan

3.2 Creating and implement Data for Routing system.

3.2.1 OSM Data.

Open Street Map was founded in the year 2004 by Steve Coast. In the beginning the project was focusing mainly on mapping United Kingdom but later in the year 2006 Open Street Map Foundation was started to encourage the growth, development and distribution of free geospatial data to the users. Open Street is one of the leading example of Volunteered Geographic Information (VGI) on the internet. In term of the data quality OSM has shown compare favorably with other source of spatial data (Mooney & Minghini, 2017; OpenStreetMap). The different data structure in OSM are:

- Nodes (Point object with geographic position)
- Ways are lists of nodes (line strings, linear rings, used for polygons as well)
- Relation are group of nodes

- Tags can be applied to the nodes.

In this study the vector data are exported from OSM web site. The data are show as in Figure 3

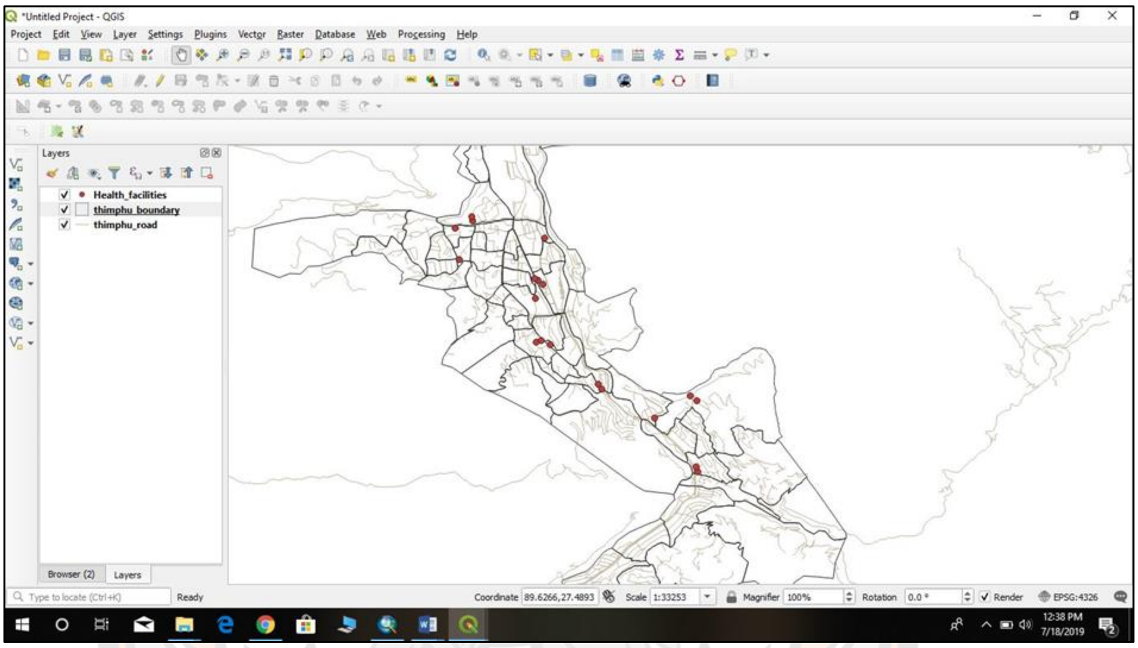


Figure 3 Open Street Map Data of the Study Area on QGIS

3.2.2 Implement road network data for Routing Application.

The road network data for this study was obtained from Open Street Map (OSM), Routing cannot be done without assigning the source, target node and the travelling cost. Between each points we need to find the shortest path but without creating a topology on the network we cannot find the shortest path. So, for this purpose we need to create the topology on the network first. Since OSM data don’t have a topology values, the data needs to process before importing to DBMS(QGIS).

For this study the OSM file is being converted to shape files and the before importing to the RDBMS. In the study road network data named as *ways* and point data as *ways_vertices_pgr.*

3.2.3 Import Road Network Data and Point Data in to Database

Before importing the SQL data to the database, we need to create a data base then add the PostGIS functions in to the database. Then after adding the PostGIS functions we need to add the routing functions by adding pgRouting in to the database. The following are the steps: -

# Creating Database

CREATE DATABASE DATABASE NAME.

CREATE DATABASE Thimphu.

# Adding PostGIS functions

CREATE EXTENSION postgis

# Adding pgRouting functions

CREATE EXTENSION pgrouting;

3.2.4 Create a Network Topology

Since the road network data downloaded from the Open Street Map does not have the network topology to implement the routing system. We need to create a network topology which can be split and snap all the nodes of the data to separate the segments in the road network. To get the network topology osm2pgrouting converter was used. osm2pgrouting is a command line tool that imports OpenStreetMap data into a pgRouting database. It builds the routing network topology automatically and creates tables for feature types and road classes. osm2pgrouting was primarily written by Daniel Wendt and is now hosted with the pgRouting project site(Project). The command line used for the osm2pgRouting (*osm2pgrouting\ -f thimphu.osm \ -d final_work \ -u user.)*

After running the command osm2pgRouting the following files are generated:

a. ways
b. ways_gid_sequence
c. ways_vertices_pgr
d. ways_vertices_pgr_id

Before we use the road network data file for the routing services we need to change the data types of the attributes of the road data (ways). The following command is used to change the data type of the road data: -

```
ALTER TABLE ways
ALTER COLUMN source TYPE BIGINT,
ALTER COLUMN target TYPE BIGINT,
ALTER COLUMN cost TYPE BIGINT,
ALTER COLUMN length TYPE BIGINT,
ALTER COLUMN reverse_cost TYPE BIGINT;
ALTER TABLE ways
ALTER COLUMN source TYPE BIGINT,
ALTER COLUMN target TYPE BIGINT,
ALTER COLUMN cost TYPE BIGINT,
ALTER COLUMN length TYPE BIGINT,
ALTER COLUMN reverse_cost TYPE BIGINT;
```

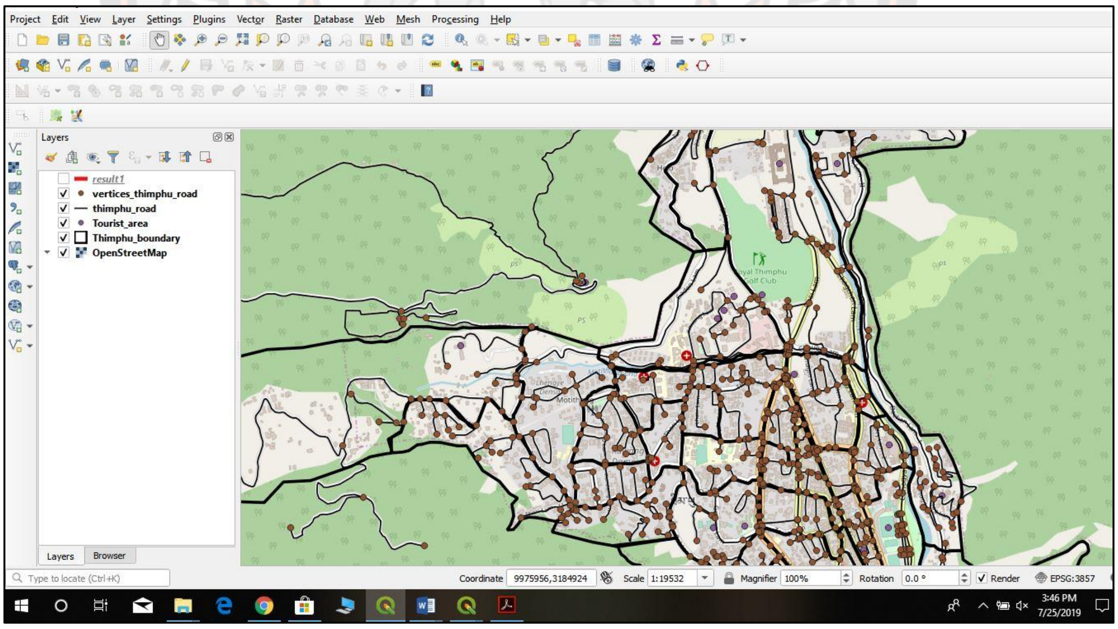


Figure 4 Showing the Nodes of Thimphu_Road network on QGIS

3.3 Shortest Path Search with pgRouting Functions

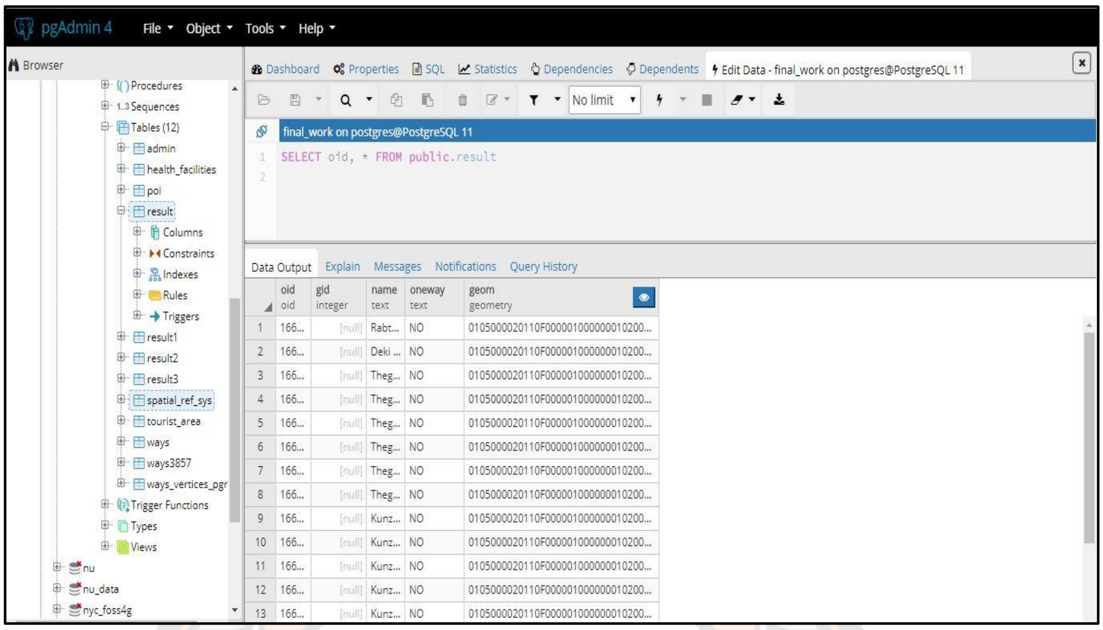
3.3.1 Static Shortest path using Dijkstra Algorithm.

Dijkstra Algorithm is the first algorithm of pgRouting which is used to find the shortest distance from Point A to Point B. For this algorithm we just need the source and target id. Dijkstra's algorithm starts from a source node, and in each iteration adds another vertex to the shortest-path spanning tree. Dijkstra care only the sum of their weights (finds the path with lowest cost) between that vertex and every other vertex. Its confirm that the Dijkstra algorithm find the shortest path from the starting of the point to end point if there is no negative cost (Distance) for the vertexes (Deepa, 2018; Road Safty and Transport Authority, June 2019).

The following is a SQL query used for creating the network topology: -

```
Drop table if exists result1;
Create table result1 (gid int4) with oids;
Select addgeometrycolumn('result1','geom',4326,'MultiLineString',2);
Insert into result1(geom)
Select geom from thimphu_road w, (
SELECT * FROM pgr_dijkstra( 'SELECT gid AS id, source, target,  length  AS cost
     FROM thimphu_road',
                712,800,
        directed := false)) as rt where w.gid=rt.edge;
```

In the above example the SQL query have find the shortest path from node 712 to 800.After the query is being called a table with gid ,name ,one way and geom(geometry) is generated in the data base as shown in figure 5 and then it can be added to QGIS as shown in Figure 6.



```
SELECT oid, * FROM public.result
```

	oid oid	gid integer	name text	oneway text	geom geometry
1	166...	[null]	Rabt...	NO	0105000020110F000001000000010200...
2	166...	[null]	Deki ...	NO	0105000020110F000001000000010200...
3	166...	[null]	Theg...	NO	0105000020110F000001000000010200...
4	166...	[null]	Theg...	NO	0105000020110F000001000000010200...
5	166...	[null]	Theg...	NO	0105000020110F000001000000010200...
6	166...	[null]	Theg...	NO	0105000020110F000001000000010200...
7	166...	[null]	Theg...	NO	0105000020110F000001000000010200...
8	166...	[null]	Theg...	NO	0105000020110F000001000000010200...
9	166...	[null]	Kunz...	NO	0105000020110F000001000000010200...
10	166...	[null]	Kunz...	NO	0105000020110F000001000000010200...
11	166...	[null]	Kunz...	NO	0105000020110F000001000000010200...
12	166...	[null]	Kunz...	NO	0105000020110F000001000000010200...
13	166...	[null]	Kunz...	NO	0105000020110F000001000000010200...

Figure 5 Showing the Result(shortest path)Table in pgAdmin 4

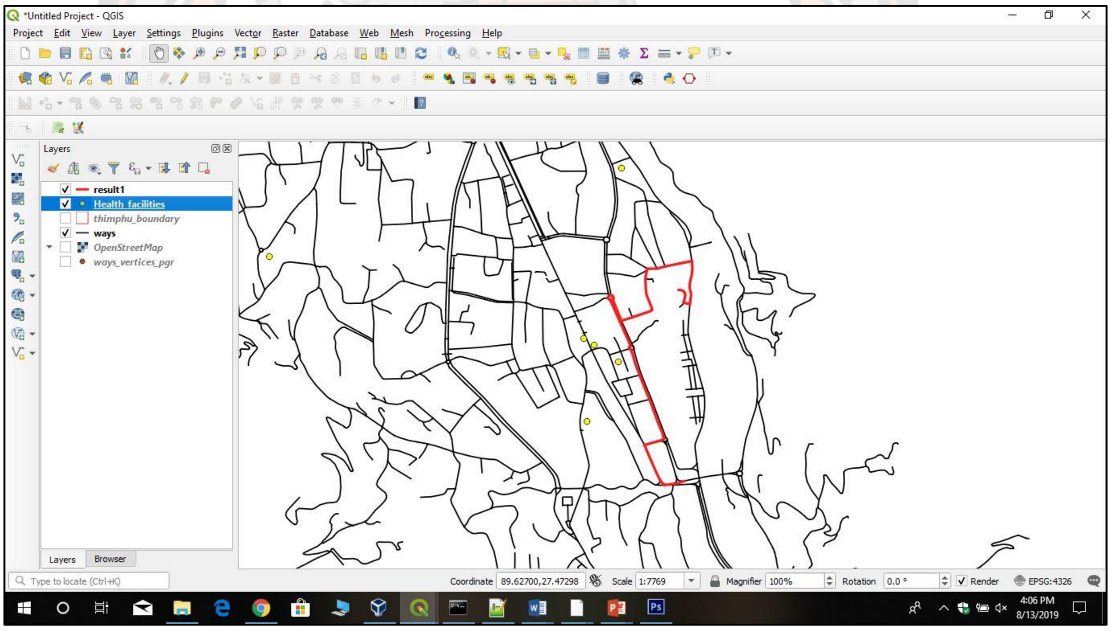


Figure 6 Showing the Shortest path of Thimphu_Road network on QGIS

3.4 Implementation of Web GIS for Dynamic Routing System

3.4.1 Modification of Road Network Data

To find the dynamic routing for the road network a dynamic cost value was created depending on both dynamic and static feature of every road segment. A dynamic cost was added to attribute column and the value was given as per the road condition in the real time, for example if there is a road block due to accident or a traffic jam the value will be changed to infinity so that the system will take the next alternative route.

The dynamic cost attribute column was created in order to keep the record of those data which are change dynamically other than the main static spatial data.

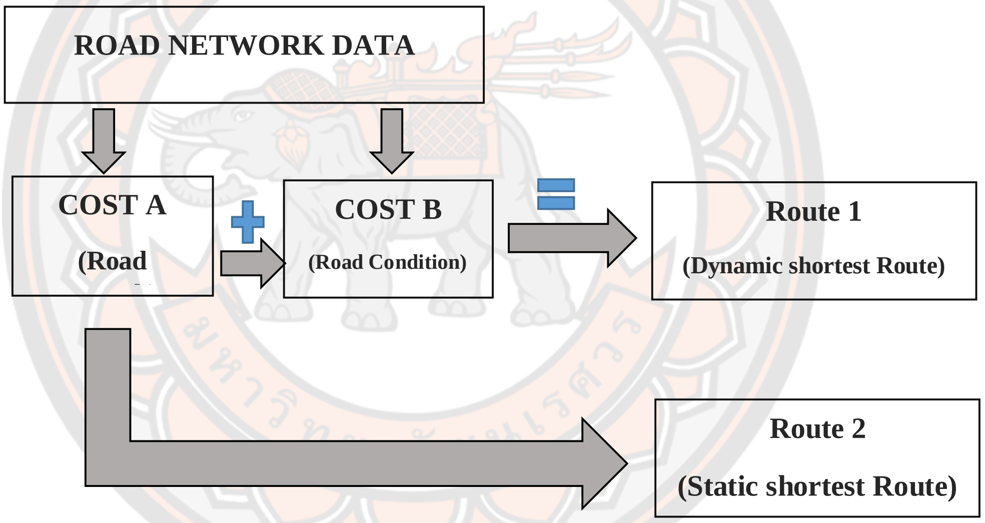
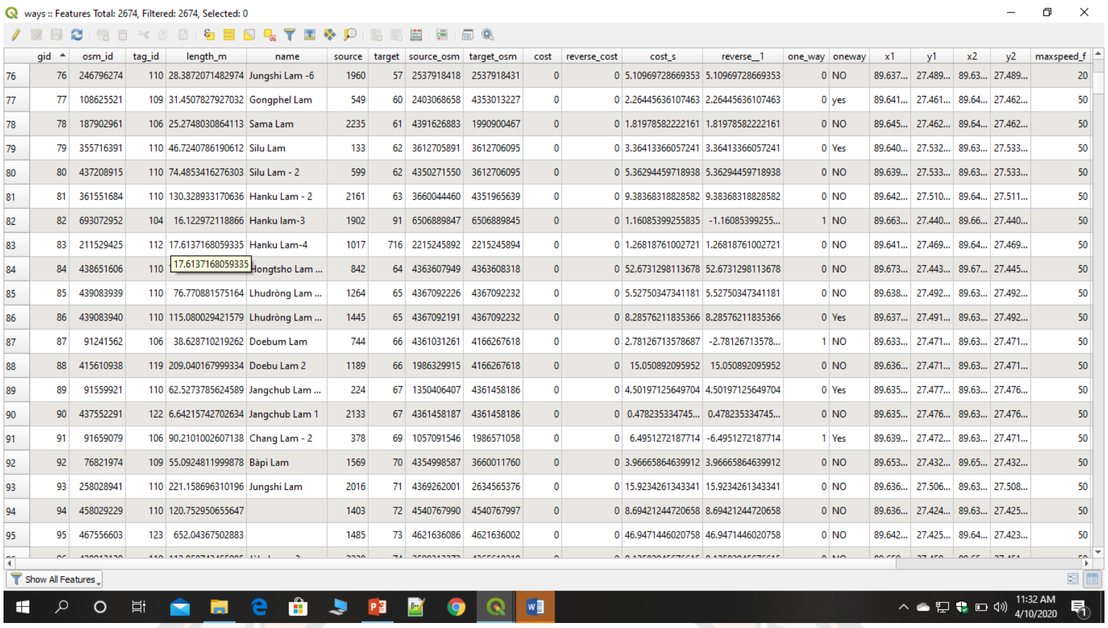


Figure 7 Showing the structure of the dynamic cost.

3.4.2 Modification of functions in pgRouting functions

pgRouting finds the best and the shortest distance from the graph. It can also be modified and enhanced to get the cost values from multiple variables cost. Distance is taken as the static cost for the road network. While in the dynamic shortest path search it will take in to the account of real road network.


ways :: Features Total: 2674, Filtered: 2674, Selected: 0

	gid	osm_id	tag_id	length_m	name	source	target	source_osm	target_osm	cost	reverse_cost	cost_s	reverse__1	one_way	oneway	x1	y1	x2	y2	maxspeed_f
76	76	246796274	110	28.3872071482974	Jungshi Lam -6	1960	57	2537918418	2537918431	0	0	5.10969728669353	5.10969728669353	0	NO	89.637...	27.489...	89.63...	27.489...	20
77	77	108625521	109	31.4507827927032	Gongphel Lam	549	60	2403068658	4353013227	0	0	2.26445636107463	2.26445636107463	0	yes	89.641...	27.461...	89.64...	27.462...	50
78	78	187902961	106	25.2748030864113	Sama Lam	2235	61	4391626883	1990900467	0	0	1.81978582222161	1.81978582222161	0	NO	89.645...	27.462...	89.64...	27.462...	50
79	79	355716391	110	46.7240786190612	Silu Lam	133	62	3612705891	3612706095	0	0	3.36413366057241	3.36413366057241	0	Yes	89.640...	27.532...	89.63...	27.533...	50
80	80	437208915	110	74.4853416276303	Silu Lam - 2	599	62	4350271550	3612706095	0	0	5.36294459718938	5.36294459718938	0	NO	89.639...	27.533...	89.63...	27.533...	50
81	81	361551684	110	130.328933170636	Hanku Lam - 2	2161	63	3660044460	4351965639	0	0	9.38368318828582	9.38368318828582	0	NO	89.642...	27.510...	89.64...	27.511...	50
82	82	693072952	104	16.122972118866	Hanku lam-3	1902	91	6506889847	6506889845	0	0	1.16085399255835	-1.16085399255...	1	NO	89.663...	27.440...	89.66...	27.440...	50
83	83	211529425	112	17.6137168059335	Hanku Lam-4	1017	716	2215245892	2215245894	0	0	1.26818761002721	1.26818761002721	0	NO	89.641...	27.469...	89.64...	27.469...	50
84	84	438651606	110	17.6137168059335	Hongtsho Lam ...	842	64	4363607949	4363608318	0	0	52.6731298113678	52.6731298113678	0	NO	89.673...	27.443...	89.67...	27.445...	50
85	85	439083939	110	76.770881575164	Lhudròng Lam ...	1264	65	4367092226	4367092232	0	0	5.52750347341181	5.52750347341181	0	NO	89.638...	27.492...	89.63...	27.492...	50
86	86	439083940	110	115.080029421579	Lhudròng Lam ...	1445	65	4367092191	4367092232	0	0	8.28576211835366	8.28576211835366	0	Yes	89.637...	27.491...	89.63...	27.492...	50
87	87	91241562	106	38.628710219262	Doebum Lam	744	66	4361031261	4166267618	0	0	2.78126713578687	-2.78126713578...	1	NO	89.633...	27.471...	89.63...	27.471...	50
88	88	415610938	119	209.040167999334	Doebu Lam 2	1189	66	1986329915	4166267618	0	0	15.050892095952	15.050892095952	0	NO	89.636...	27.471...	89.63...	27.471...	50
89	89	91559921	110	62.5273785624589	Jangchub Lam ...	224	67	1350406407	4361458186	0	0	4.50197125649704	4.50197125649704	0	Yes	89.635...	27.477...	89.63...	27.476...	50
90	90	437552291	122	6.64215742702634	Jangchub Lam 1	2133	67	4361458187	4361458186	0	0	0.478235334745...	0.478235334745...	0	NO	89.635...	27.476...	89.63...	27.476...	50
91	91	91659079	106	90.2101002607138	Chang Lam - 2	378	69	1057091546	1986571058	0	0	6.4951272187714	-6.4951272187714	1	Yes	89.639...	27.472...	89.63...	27.471...	50
92	92	76821974	109	55.0924811999878	Bàpi Lam	1569	70	4354998587	3660011760	0	0	3.96665864639912	3.96665864639912	0	NO	89.653...	27.432...	89.65...	27.432...	50
93	93	258028941	110	221.158696310196	Jungshi Lam	2016	71	4369262001	2634565376	0	0	15.9234261343341	15.9234261343341	0	NO	89.636...	27.506...	89.63...	27.508...	50
94	94	458029229	110	120.752950655647		1403	72	4540767990	4540767997	0	0	8.69421244720658	8.69421244720658	0	NO	89.636...	27.424...	89.63...	27.425...	50
95	95	467556603	123	652.04367502883		1485	73	4621636086	4621636002	0	0	46.9471446020758	46.9471446020758	0	NO	89.642...	27.425...	89.64...	27.423...	50

Figure 8 Attribute table of road network data.

3.4.3 Modification of functions for Dynamic Distance

The shortest path function was modified according to the road condition. This function was modified in pgr_dijkstra function under one of the libraries in pgRouting. To search for the dynamic shortest route, the function needs to modify the value of Cost A as Cost A + Cost B. Cost A is the distance of the road segment and Cost B is the road condition set by the system administrator. Usually the value for Cost A will be a static value and Cost B will be dynamic. The value for the Cost B will change if there is a temporary road block due to car accident or traffic jam.

The command used to modify the Dynamic shortest route in pgRouting function is as show below: -

update ways set dynamic_cost= 1000000 where gid= 1573;

update ways set dynamic_cost= length where gid= 1573;

drop table if exists result;

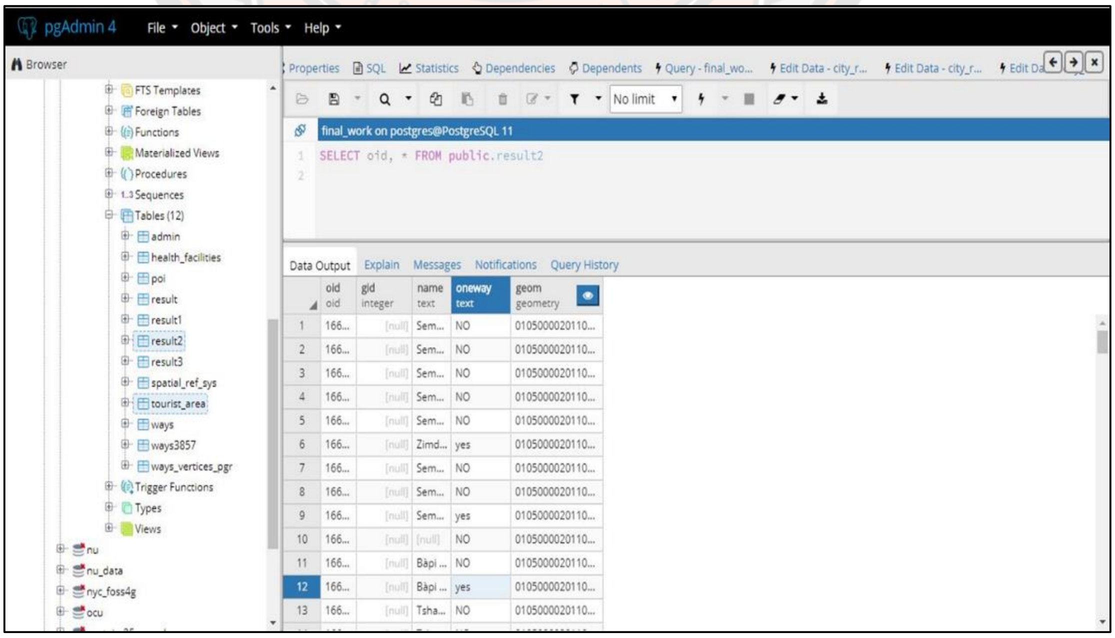
create table result (gid int4, name text, oneway text) with oids;

select addgeometrycolumn('result','geom',3857,'MULTILINESTRING',2);

insert into result(geom, name, oneway)

```
select geom, name,oneway
from ways w,(
            SELECT * FROM pgr_dijkstra(
    'SELECT gid as id,
         source,
         target,
         length + dynamic_cost AS cost
        FROM ways',
     33,1979,
    directed := false) )as rt
    where w.gid=rt.edge;
```

In the above code the Cost A= Length of the road and Cost B = Dynamic cost which is assigned by the system administrator in the data base. After running the query in pgAdmin as shown in figure 8 ,a table will be generated in the data base and which is further taken in to QGIS. Figure 9 shows the shortest distance from point A to Point B, but when we assign a value as infinity in dynamic cost for gid 1573 the road gets blocked as shown in figure 10, and when we run the query again then we get the new alternative route as shown in figure 11.



```
SELECT oid, * FROM public.result2
```

	oid oid	gid integer	name text	oneway text	geom geometry
1	166...	[null]	Sem...	NO	0105000020110...
2	166...	[null]	Sem...	NO	0105000020110...
3	166...	[null]	Sem...	NO	0105000020110...
4	166...	[null]	Sem...	NO	0105000020110...
5	166...	[null]	Sem...	NO	0105000020110...
6	166...	[null]	Zimd...	yes	0105000020110...
7	166...	[null]	Sem...	NO	0105000020110...
8	166...	[null]	Sem...	NO	0105000020110...
9	166...	[null]	Sem...	yes	0105000020110...
10	166...	[null]	[null]	NO	0105000020110...
11	166...	[null]	Bàpi ...	NO	0105000020110...
12	166...	[null]	Bàpi ...	yes	0105000020110...
13	166...	[null]	Tsha...	NO	0105000020110...

Figure 9 Showing the Dynamic Shortest path Table in pgAdmin 4

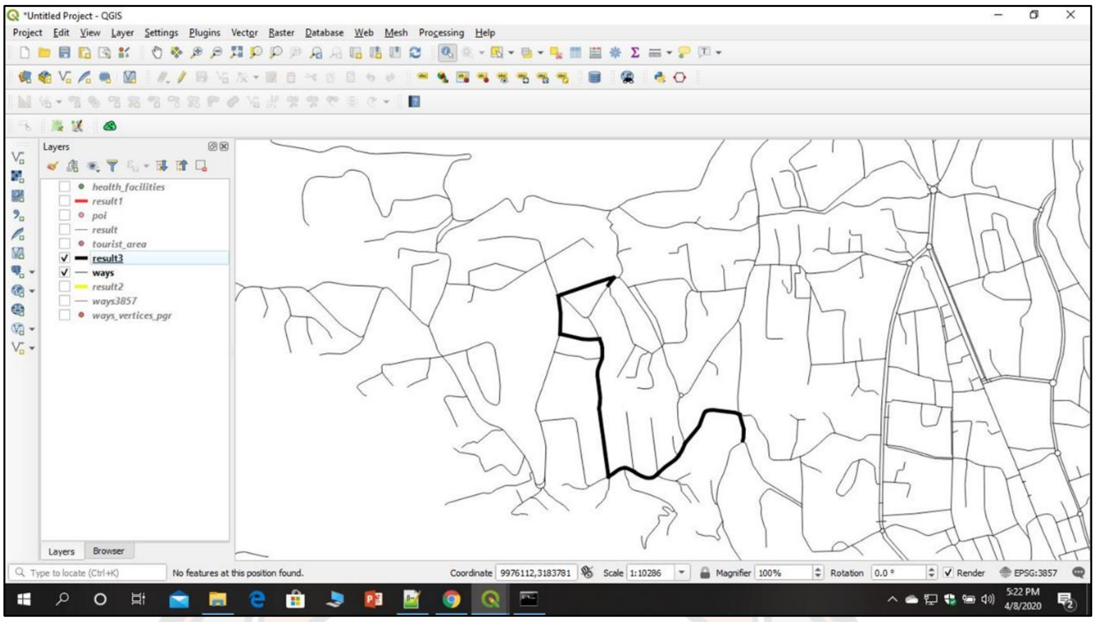


Figure 10 Showing the Shortest path on QGIS

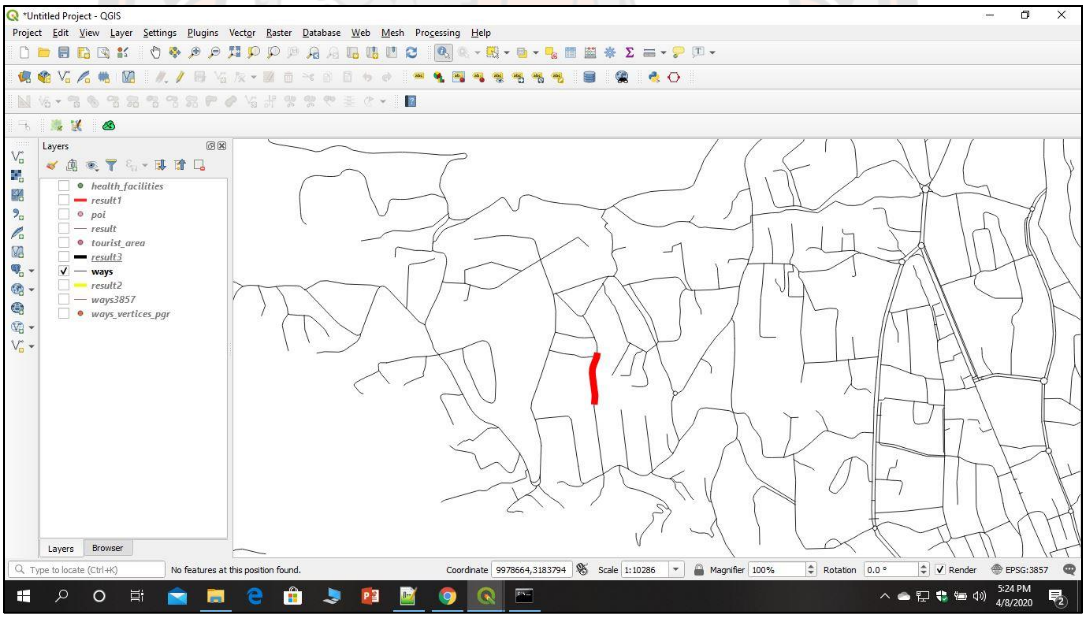


Figure 11 Showing a road block for gid 1573 in QGIS

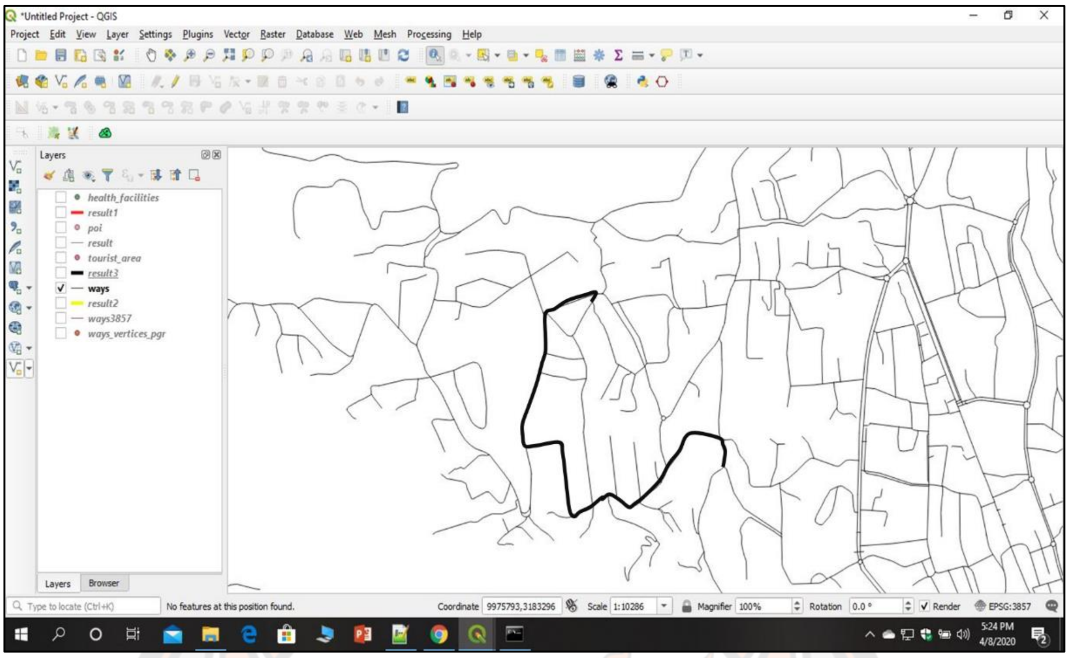


Figure 12 Showing a Dynamic shortest route from point A to Point B

3.5 Development of web Application for the Routing Services.

The system was fully developed by using FOSS4G software's as shown in figure 14. Figure 15 shows the dynamic routing framework. PgRouting software was used to do the query for the road data from the road database. When the information about the road network is received it's being updated in the database and new route is being generated. Table 1 shows the detail information about the software used.

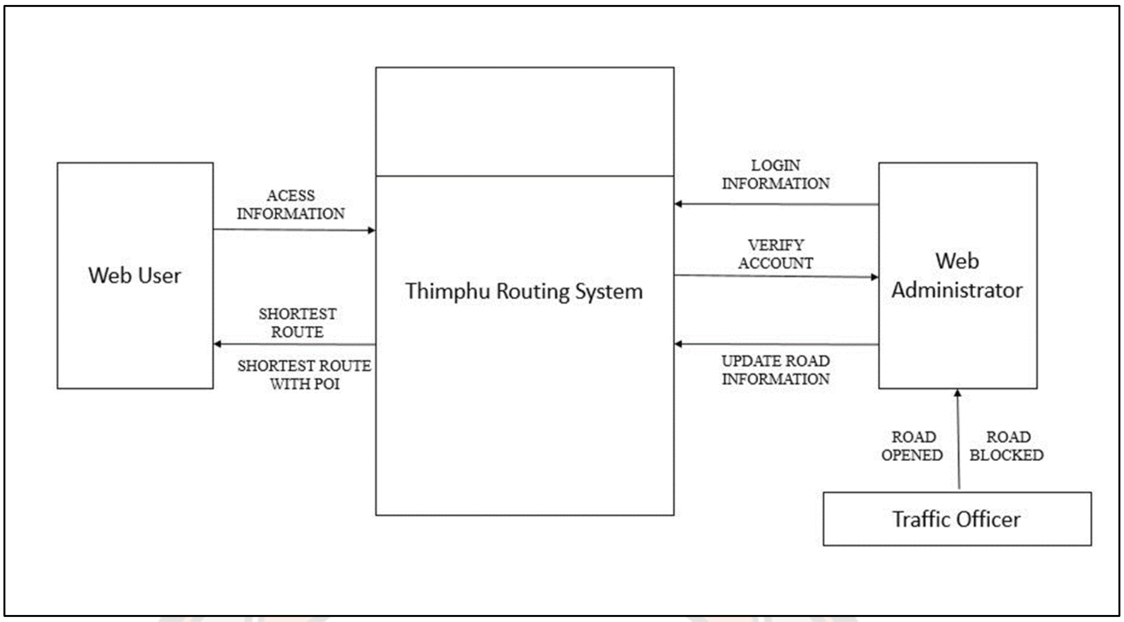


Figure 13 Context Diagram of Thimphu Routing System

3.5.1 System Architecture

Web GIS is a process of designing, implementing, generating and delivering maps on the World Wide Web. GIS integrates and relates data with spatial components and support users to view the data in the form of maps making decision making easier through visualization. The use of GIS functionality through internet has become a powerful tool for planning and people's participation in planning.

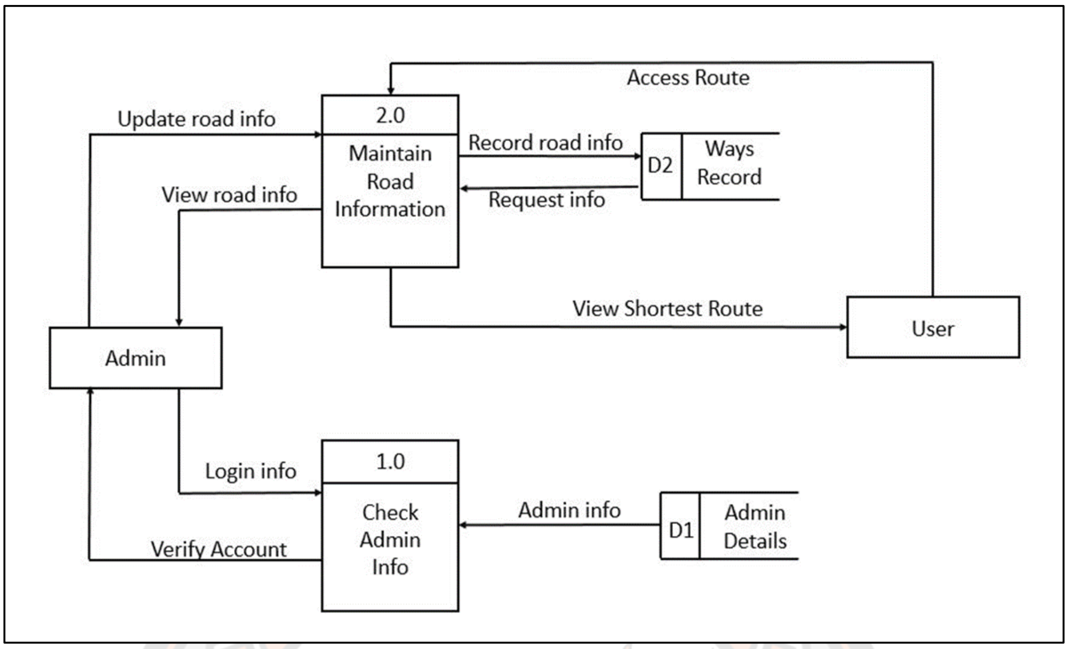


Figure 14 Data Flow Diagram of Thimphu Routing System

The architecture of web GIS has three layers namely the interface layer, application layer and the data base layer. The client accesses the system through the interface layer which takes the input and then shows the output on the system. Application layer shows the map visualization functions like panning, zooming etc. and even other functions like selection, editing and querying etc. Database layer handles the data and how it is being presented and distributed. The system Architecture is as shown as figure 15.

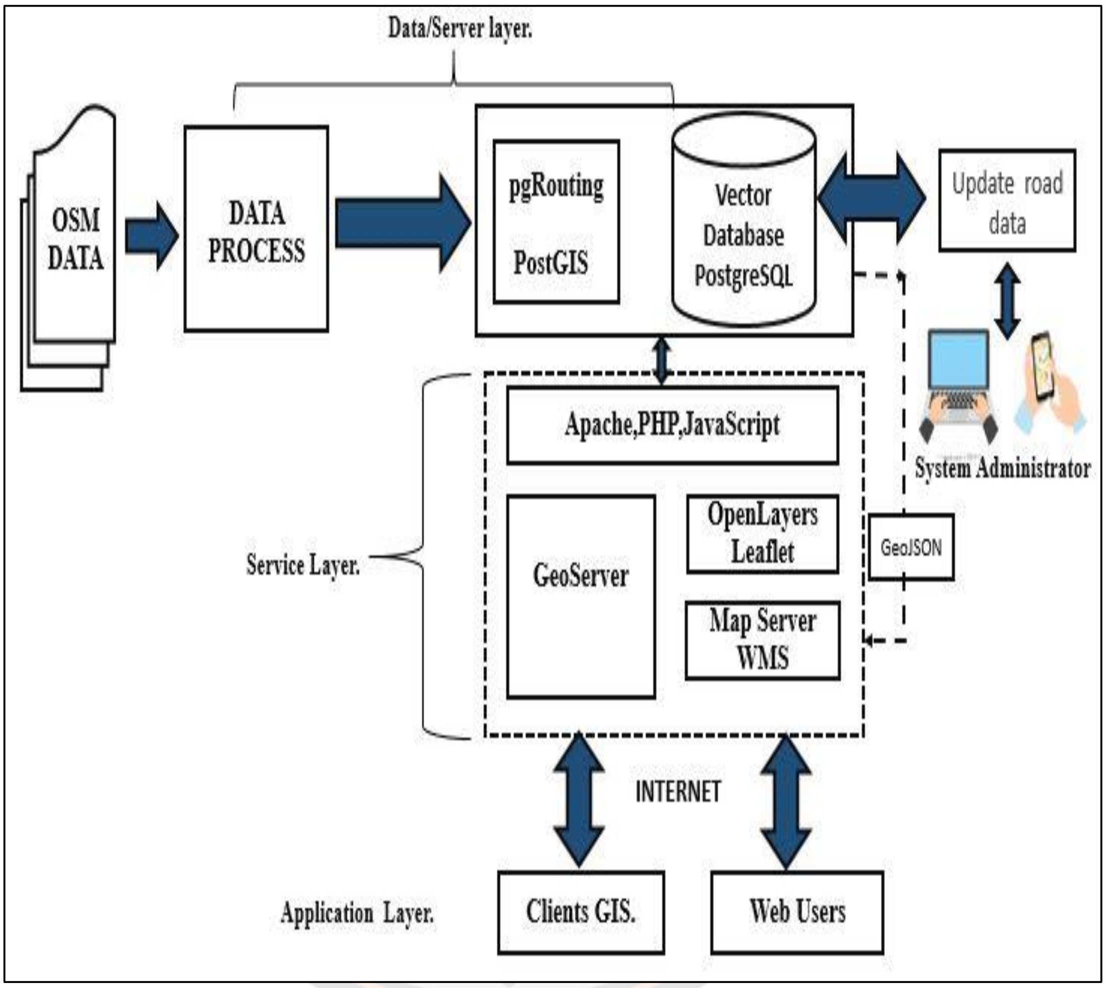


Figure 15 The System Framework

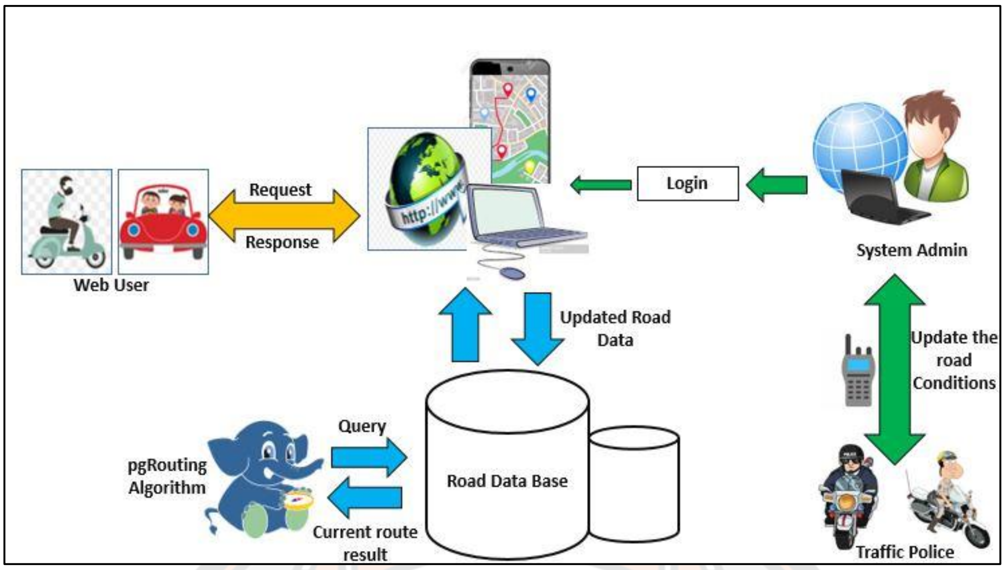


Figure 16 Workflow of a Dynamic Routing System

Table 1 Software Details

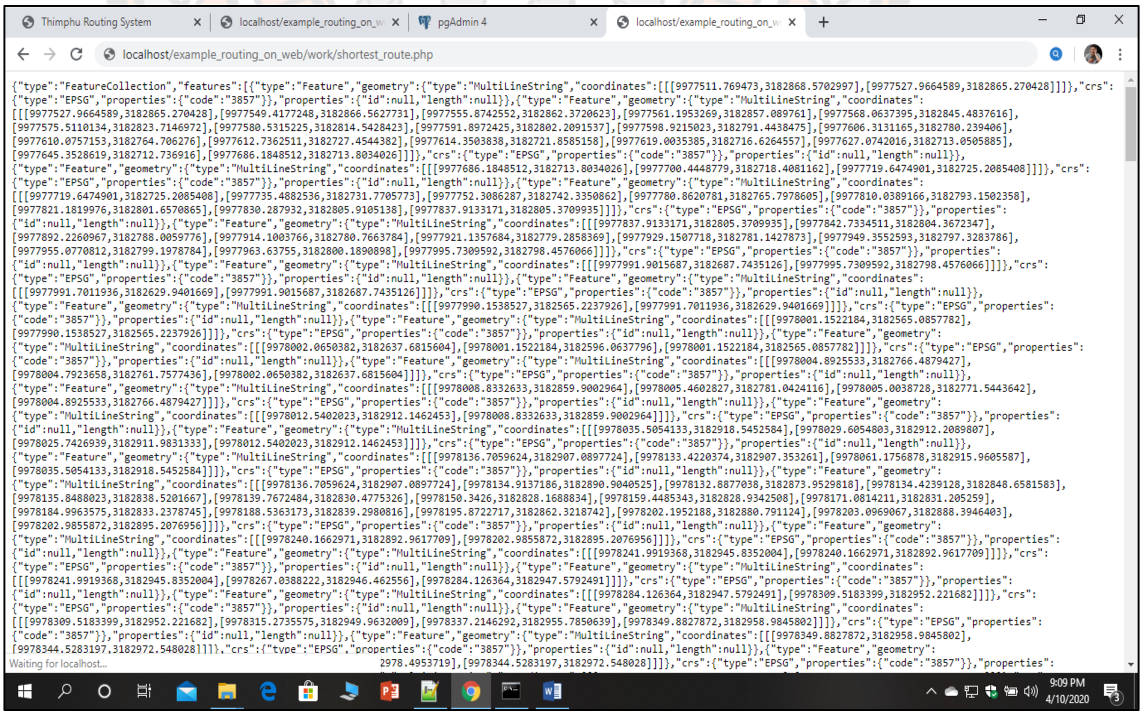
Software	**Version**	**Functions**
PostgresSQL	11.1	Main database (RDBMs) Spatial and non-Spatial data are stored
PostGIS	2.5.1	Spatial Data extension for PostgreSQL.
QGIS	3.8	Download OSM data Converting and test the result
pgRouting	2.6.2	Network analysis
Geoserver	2.12.4	Visualization /Web based client mapping application(Web Map Service)
OpenLayer	6.0.1	For front end and loading the map in the server
Google Chrome	77.0.3865	Web browser
Java Script and PHP	1.8.5 and 7.2	Script language

3.5.3 Creating a PHP script for Routing Services.

For this study coding was mainly done in PHP to make the routing query and send back the result in the web clients as GeoJSON format which shows the map result on the Web Interface.

The routing PHP scripts following the following steps (As shown in Appendix 2)

1. Gets the information about the start and end point coordinates by clicking on the map by the user.
2. It finds the closest road segment of the starting and ending point.
3. Takes the starting node of the road segment as a source and it takes the ending node as the target to show the route respectively.
4. Upon choosing the type of function it shows the shortest route by querying the data base.
5. The query result as GeoJSON will be send back to the web client


Thimphu Routing System
localhost/example_routing_on_w
pgAdmin 4
localhost/example_routing_on_w
localhost/example_routing_on_web/work/shortest_route.php
Waiting for localhost...
9:09 PM
4/10/2020

```
{"type":"FeatureCollection","features":[{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9977511.769473,3182868.5702997],[9977527.9664589,3182865.270428]]]},"crs":
{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":
[[[9977527.9664589,3182865.270428],[9977549.4177248,3182866.5627731],[9977555.8742552,3182862.3720623],[9977561.1953269,3182857.089761],[9977568.0637395,3182845.4837616],
[9977575.5110134,3182823.7146972],[9977580.5315225,3182814.5428423],[9977591.8972425,3182802.2091537],[9977598.9215023,3182791.4438475],[9977606.3131165,3182780.239406],
[9977610.0757153,3182764.706276],[9977612.7362511,3182727.4544382],[9977614.3503838,3182721.8585158],[9977619.0035385,3182716.6264557],[9977627.0742016,3182713.0505885],
[9977645.3528619,3182712.736916],[9977686.1848512,3182713.8034026]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},
{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9977686.1848512,3182713.8034026],[9977700.4448779,3182718.4081162],[9977719.6474901,3182725.2085408]]]},"crs":
{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":
[[[9977719.6474901,3182725.2085408],[9977735.4882536,3182731.7705773],[9977752.3086287,3182742.3350862],[9977780.8620781,3182765.7978605],[9977810.0389166,3182793.1502358],
[9977821.1819976,3182801.6570865],[9977830.287932,3182805.9105138],[9977837.9133171,3182805.3709935]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":
{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9977837.9133171,3182805.3709935],[9977842.7334511,3182804.3672347],
[9977892.2260967,3182788.0059776],[9977914.1003766,3182780.7663784],[9977921.1357684,3182779.2858369],[9977929.1507718,3182781.1427873],[9977949.3552593,3182797.3283786],
[9977955.0770812,3182799.1978784],[9977963.63755,3182800.1890898],[9977995.7309592,3182798.4576066]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":
{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9977991.9015687,3182687.7435126],[9977995.7309592,3182798.4576066]]]},"crs":
{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":
[[[9977991.7011936,3182629.9401669],[9977991.9015687,3182687.7435126]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},
{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9977990.1538527,3182565.2237926],[9977991.7011936,3182629.9401669]]]},"crs":{"type":"EPSG","properties":
{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9978001.1522184,3182565.0857782],
[9977990.1538527,3182565.2237926]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":
{"type":"MultiLineString","coordinates":[[[9978002.0650382,3182637.6815604],[9978001.1522184,3182596.0637796],[9978001.1522184,3182565.0857782]]]},"crs":{"type":"EPSG","properties":
{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9978004.8925533,3182766.4879427],
[9978004.7923658,3182761.7577436],[9978002.0650382,3182637.6815604]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},
{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9978008.8332633,3182859.9002964],[9978005.4602827,3182781.0424116],[9978005.0038728,3182771.5443642],
[9978004.8925533,3182766.4879427]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":
{"type":"MultiLineString","coordinates":[[[9978012.5402023,3182912.1462453],[9978008.8332633,3182859.9002964]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":
{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9978035.5054133,3182918.5452584],[9978029.6054803,3182912.2089807],
[9978025.7426939,3182911.9831333],[9978012.5402023,3182912.1462453]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},
{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9978136.7059624,3182907.0897724],[9978133.4220374,3182907.353261],[9978061.1756878,3182915.9605587],
[9978035.5054133,3182918.5452584]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":
{"type":"MultiLineString","coordinates":[[[9978136.7059624,3182907.0897724],[9978134.9137186,3182890.9040525],[9978132.8877038,3182873.9529818],[9978134.4239128,3182848.6581583],
[9978135.8488023,3182838.5201667],[9978139.7672484,3182830.4775326],[9978150.3426,3182828.1688834],[9978159.4485343,3182828.9342508],[9978171.0814211,3182831.205259],
[9978184.9963575,3182833.2378745],[9978188.5363173,3182839.2980816],[9978195.8722717,3182862.3218742],[9978202.1952188,3182880.791124],[9978203.0969067,3182888.3946403],
[9978202.9855872,3182895.2076956]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":
{"type":"MultiLineString","coordinates":[[[9978240.1662971,3182892.9617709],[9978202.9855872,3182895.2076956]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":
{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9978241.9919368,3182945.8352004],[9978240.1662971,3182892.9617709]]]},"crs":
{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":
[[[9978241.9919368,3182945.8352004],[9978267.0388222,3182946.462556],[9978284.126364,3182947.5792491]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":
{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9978284.126364,3182947.5792491],[9978309.5183399,3182952.221682]]]},"crs":
{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":
[[[9978309.5183399,3182952.221682],[9978315.2735575,3182949.9632009],[9978337.2146292,3182955.7850639],[9978349.8827872,3182958.9845802]]]},"crs":{"type":"EPSG","properties":
{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":{"type":"MultiLineString","coordinates":[[[9978349.8827872,3182958.9845802],
[9978344.5283197,3182972.548028]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":{"id":null,"length":null}},{"type":"Feature","geometry":
2978.4953719],[9978344.5283197,3182972.548028]]]},"crs":{"type":"EPSG","properties":{"code":"3857"}},"properties":
```

Figure 17 Sample of GeoJSON format returned to Web Clients.

The detail of GeoJSON data in Figure 17 is returned to a set of road number, coordinate as WKT and length of each road segment result. Figure 18 shows the flowchart of computation of the route on the Web Interface.

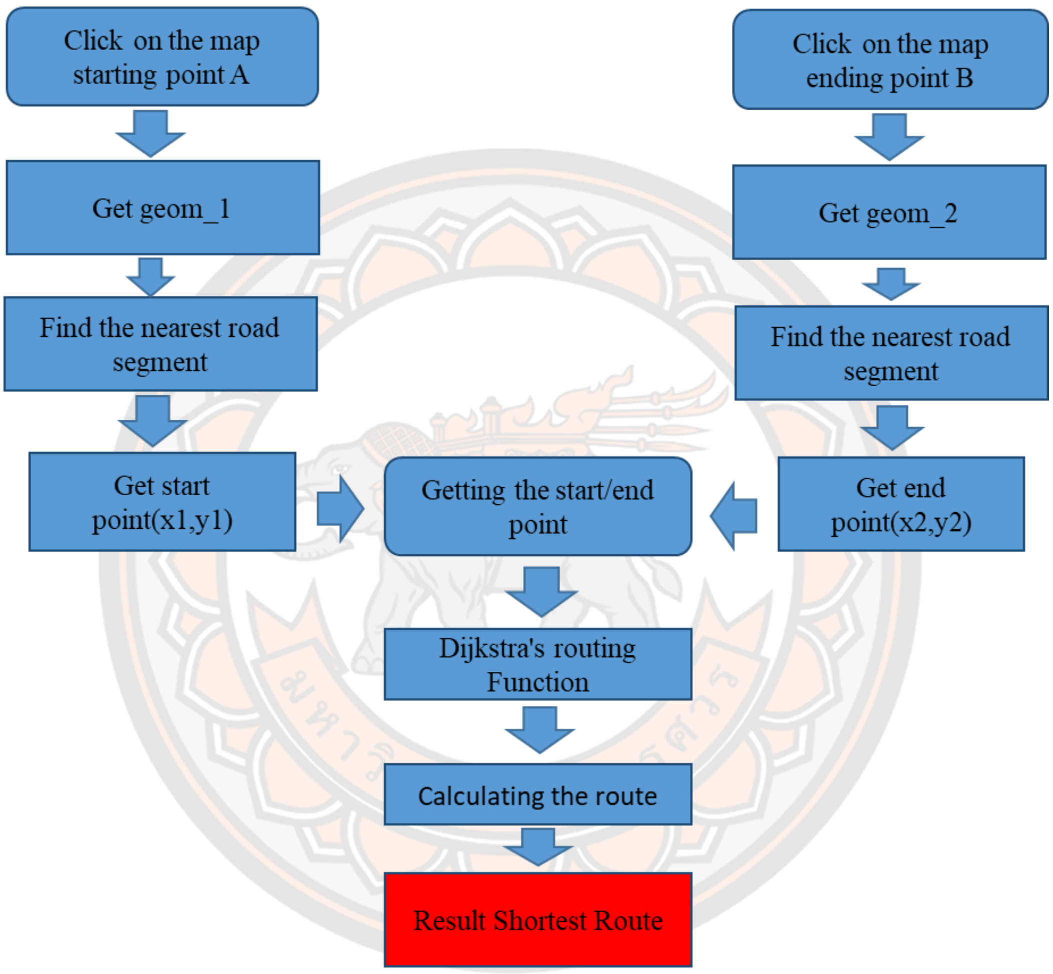

Click on the map starting point A
Get geom_1
Find the nearest road segment
Get start point(x1,y1)
Click on the map ending point B
Get geom_2
Find the nearest road segment
Get end point(x2,y2)
Getting the start/end point
Dijkstra's routing Function
Calculating the route
Result Shortest Route

Figure 18 Flow chart showing the routing process.

3.5.4 Creating a PHP script for Point of Interest (POI) along the shortest route

To find the Point of Interest (POI) along the shortest route the coding was mainly done in PHP. Firstly, a routing function was called to find the shortest route and then a function intersect was called by giving a buffer of 300 meter and

500 meters respectively for health facilities and tourist area. The result was sent back in the web clients as GeoJSON format which shows the map result on the Web Interface.

The routing PHP scripts following the following steps (As shown in Appendix 3 and Appendix 4).

a) Gets the information about the start and end point coordinates by clicking on the map by the user.
b) It finds the closest road segment of the starting and ending point.
c) Takes the starting node of the road segment as a source and it takes the ending node as the target to show the route respectively.
d) Upon choosing the type of function it shows the shortest route by querying the data base.
e) After getting the shortest route then it calls a query to intersect (st_intersects) by giving a buffer (st_buffer) of 300m for health facilities and 500 m for tourist area.
f) The query result as GeoJSON will be send back to the web client

The detail of the works is shown in the form of flow chart in figure 10 as shown below:

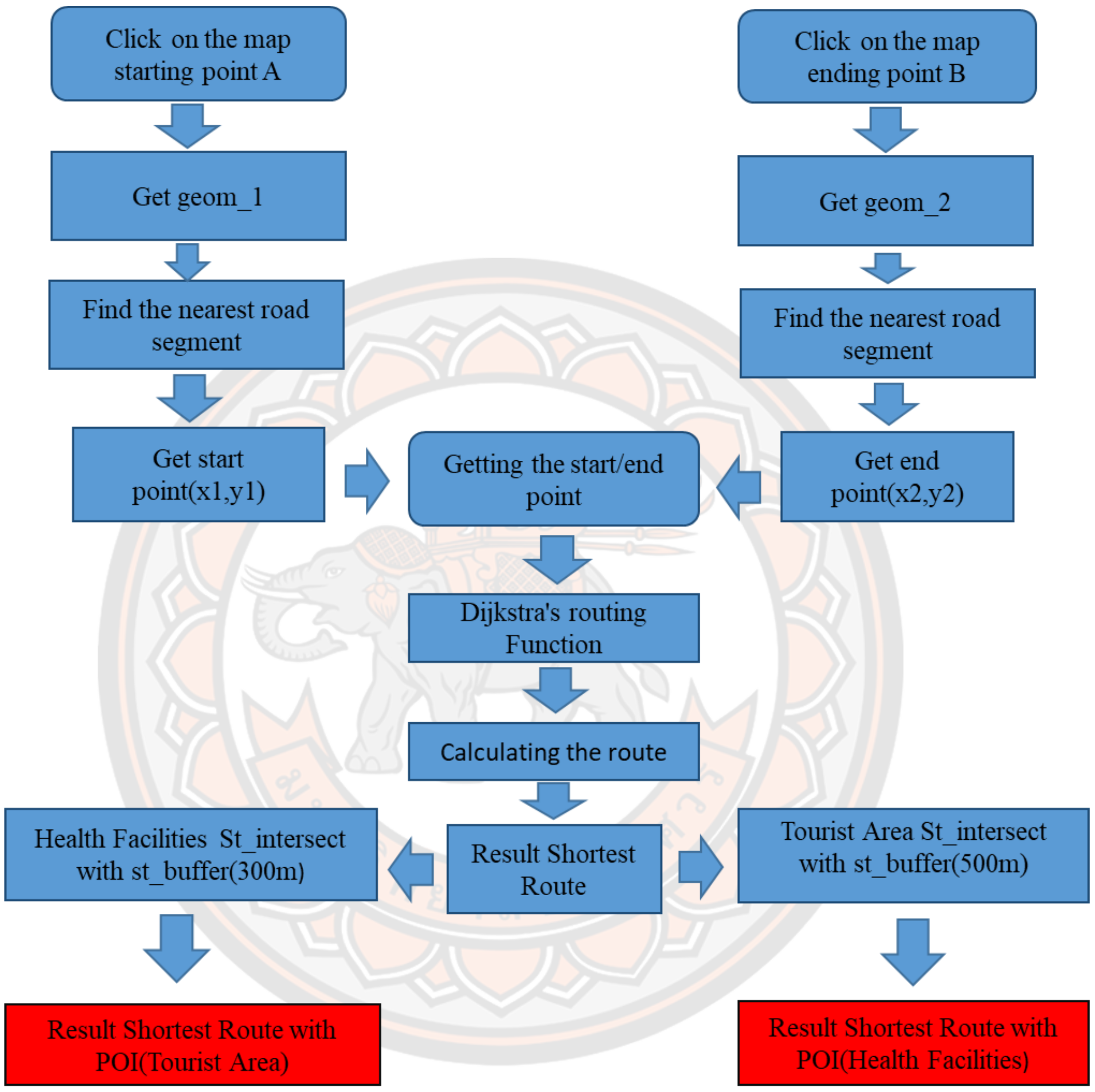

Click on the map starting point A
Get geom_1
Find the nearest road segment
Get start point(x1,y1)
Click on the map ending point B
Get geom_2
Find the nearest road segment
Get end point(x2,y2)
Getting the start/end point
Dijkstra's routing Function
Calculating the route
Result Shortest Route
Health Facilities St_intersect with st_buffer(300m)
Tourist Area St_intersect with st_buffer(500m)
Result Shortest Route with POI(Tourist Area)
Result Shortest Route with POI(Health Facilities)

Figure 19 Flow chart showing the routing process and Point of interest (POI)

3.6 Development of Web System Interfaces

The Web Map interface was developed by using OpenLayer and leaflet. Both OpenLayers and leaflet makes it easy to put a dynamic map in any web page. It can also display map tiles, vector data and markers loaded from any source. OpenLayers has been developed to further the use of geographic information of all kinds. Therefore, to develop the web interface java script and PHP code have been used as shown in.

PHP and JavaScript programming language are used to display in the browser which can be accessed via the internet network,it is an open source software and can also be embedded into HTML(OpenLayers) .GeoServer works for viewing the detail information and results for the system. The Web Map Interface has the following structure for the webpage.

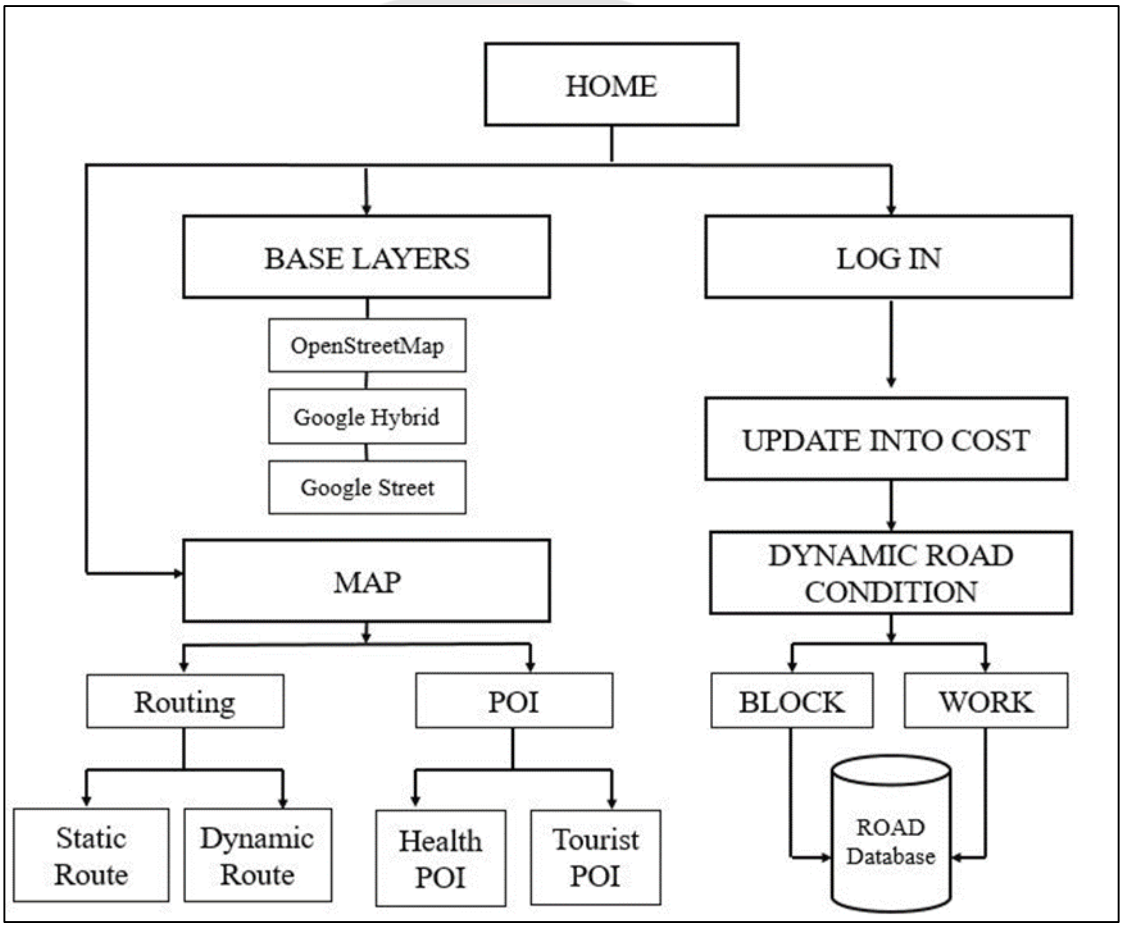


Figure 20 Structure diagram of Web Map Interface page.

3.7 Updating of Client/Administrator Interfaces.

This web interface is to update the road information. The login page for the system administrator and after logging in the page, we get a new page as showed in which shows how a system administrator/web client updates the road information.

If there is a temporary road block, the dynamic cost value should be increased to infinity. By seeing the map on the web, the administrator identifies the GID of the particular road for road closed by clicking on the road block (Leaflet).

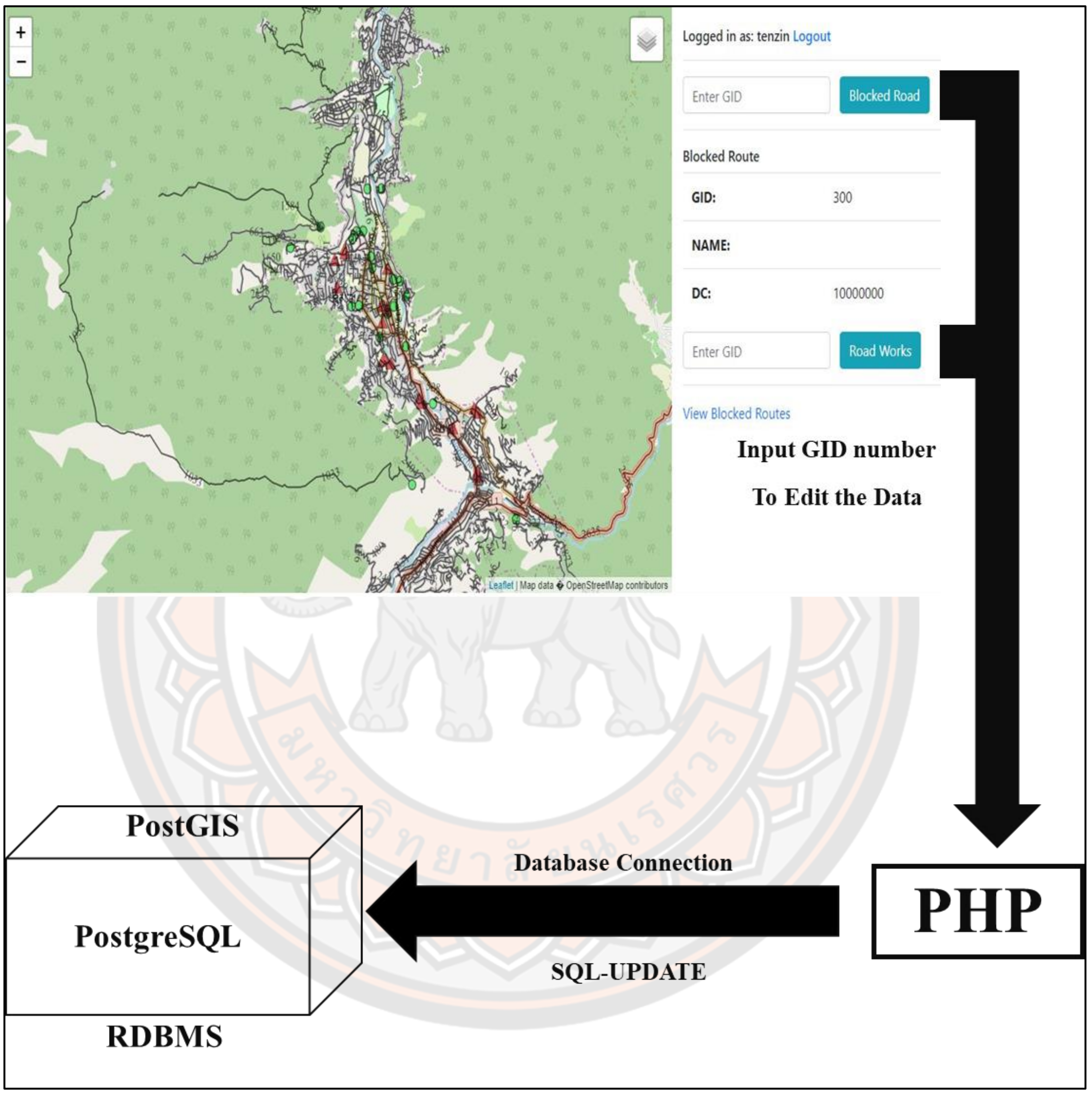


Figure 21 Updating the road condition for the Dynamic Routing

CHAPTER IV

RESULTS

4.1 Result

4.1.1 Web User Interface

The entire web user interface was designed by using PHP,HTML and OpenLayer . The user interface shows two kinds or routes, the static route and the dynamic route for the user and even the Point of Interest along the shortest route. Figure 21 shows the main web user interface used by the tourist visiting the study area, Figure 22 shows the shortest distance from Point A to point B. This function will take to the account which way is the shortest distance from Point A to Point B.

Figure 23 shows the client-side application where the condition of the road is updated for a dynamic route. For example, if there is temporary road blocked due to some accident or traffic congestion we need to find the road segment number and update on the system through the administrator interface. After updating in the data base, it will show an alternative route.

Figure 24 shows the Point of Interest (POI) for the health facilities along the shortest route. Showing the health facilities along the shortest route will be very useful during the emergencies.

Figure 25 shows the Point of Interest (POI) for the tourist areas along the shortest route in Thimphu. Showing the tourist areas along the shortest route will help the tourist to visit the places to minimize the time for their travel.

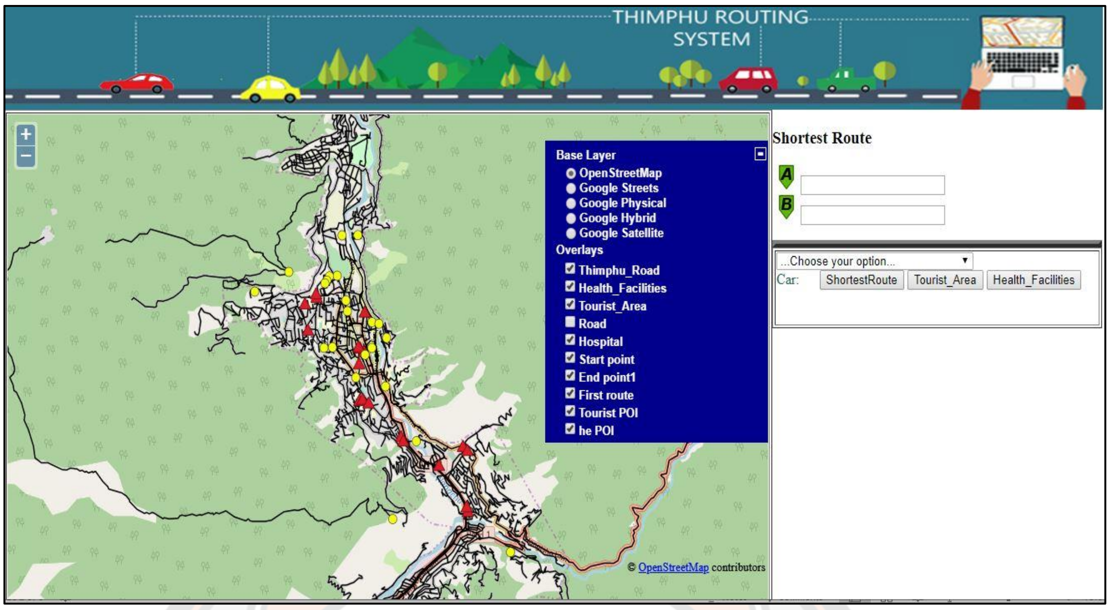


Figure 22 Showing the Web Based interface of the routing system

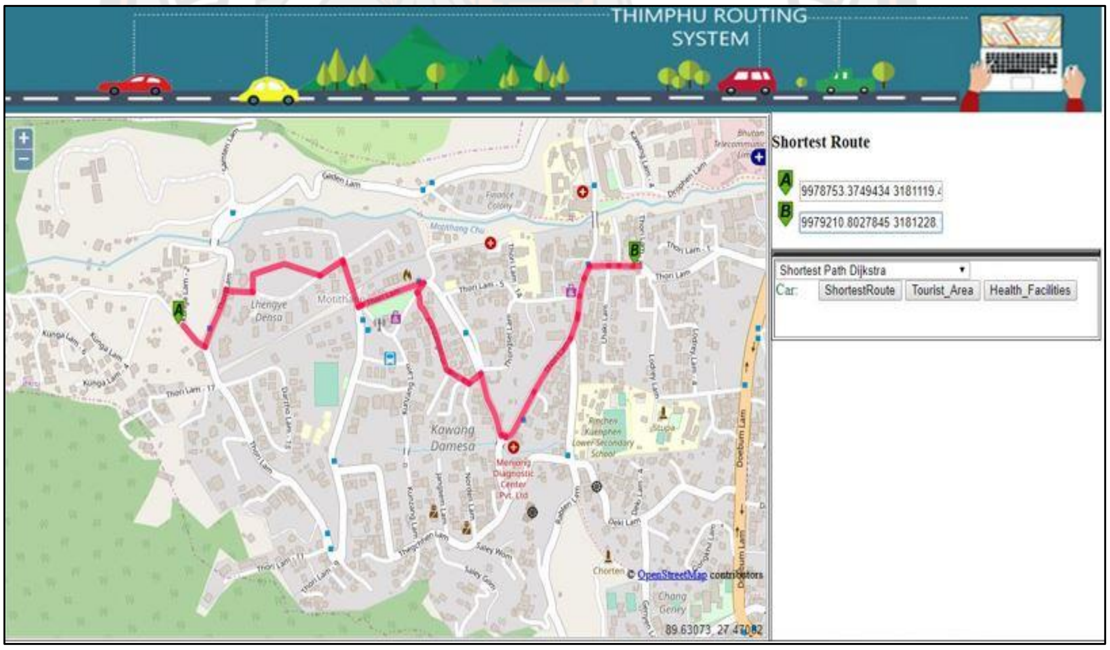


Figure 23 Web application showing the shortest distance from Point A to Point B

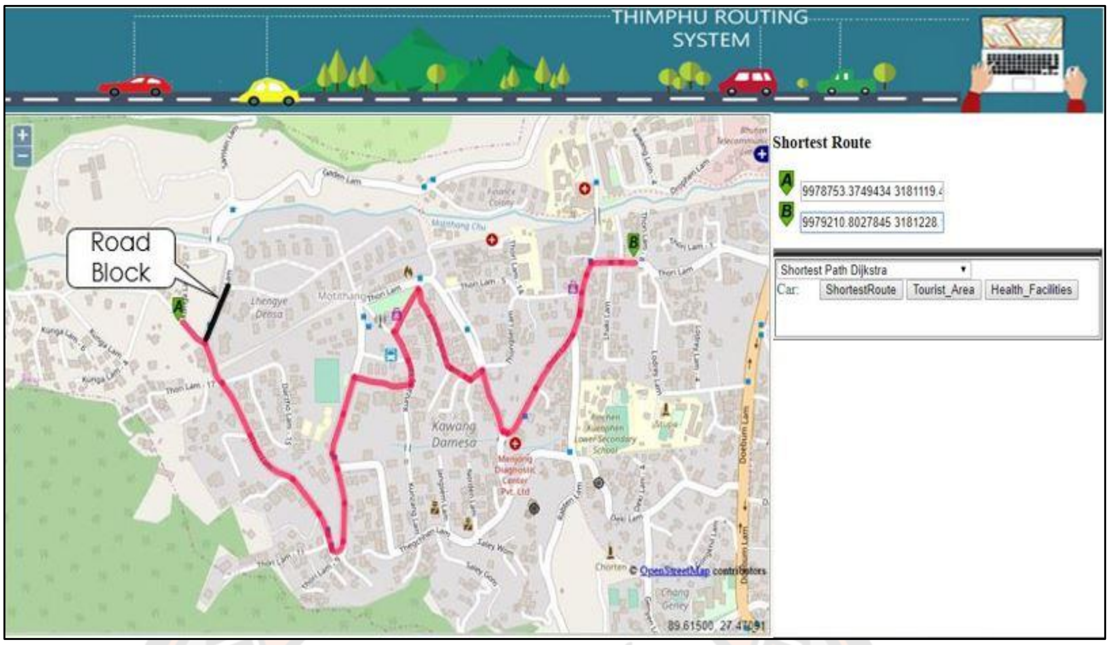
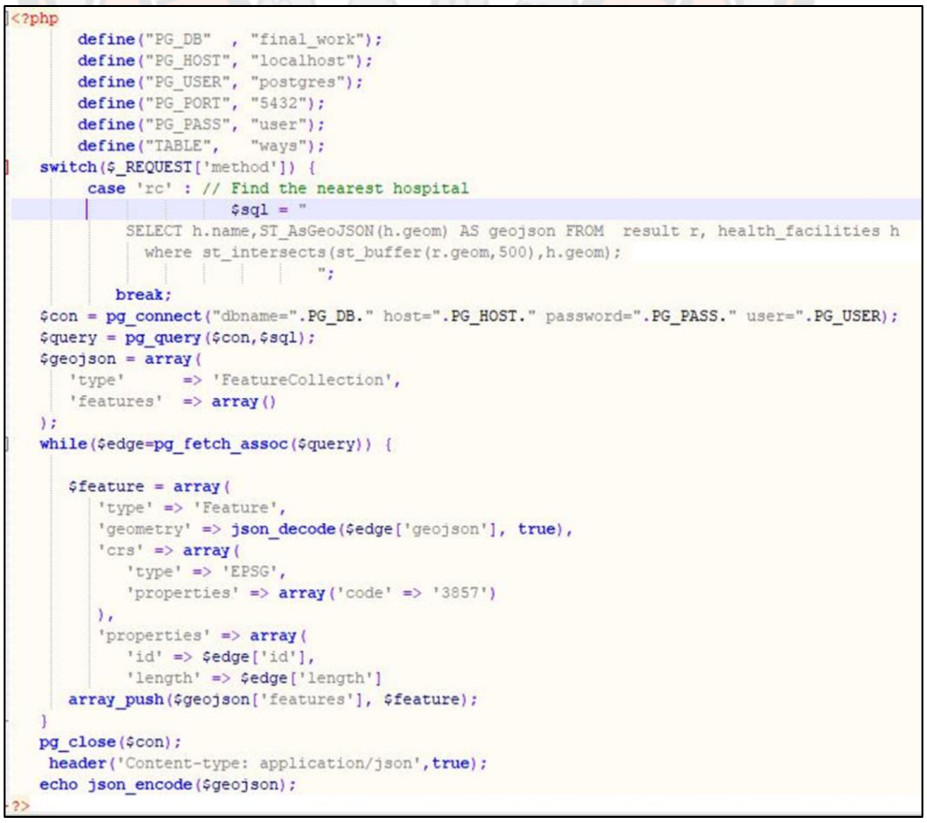


Figure 24 Web application showing the Dynamic shortest distance from Point A to Point B where there is a road block

The following is the code use to find the point of Interest for the health facilities:



```php
<?php
      define("PG_DB"  , "final_work");
      define("PG_HOST", "localhost");
      define("PG_USER", "postgres");
      define("PG_PORT", "5432");
      define("PG_PASS", "user");
      define("TABLE",   "ways");
   switch($_REQUEST['method']) {
        case 'rc' : // Find the nearest hospital
                     $sql = "
            SELECT h.name,ST_AsGeoJSON(h.geom) AS geojson FROM  result r, health_facilities h
              where st_intersects(st_buffer(r.geom,500),h.geom);
                             ";
           break;
   $con = pg_connect("dbname=".PG_DB." host=".PG_HOST." password=".PG_PASS." user=".PG_USER);
   $query = pg_query($con,$sql);
   $geojson = array(
      'type'       => 'FeatureCollection',
      'features'   => array()
   );
   while($edge=pg_fetch_assoc($query)) {

      $feature = array(
         'type' => 'Feature',
         'geometry' => json_decode($edge['geojson'], true),
         'crs' => array(
            'type' => 'EPSG',
            'properties' => array('code' => '3857')
         ),
         'properties' => array(
            'id' => $edge['id'],
            'length' => $edge['length']
      array_push($geojson['features'], $feature);
   }
   pg_close($con);
    header('Content-type: application/json',true);
   echo json_encode($geojson);
?>
```

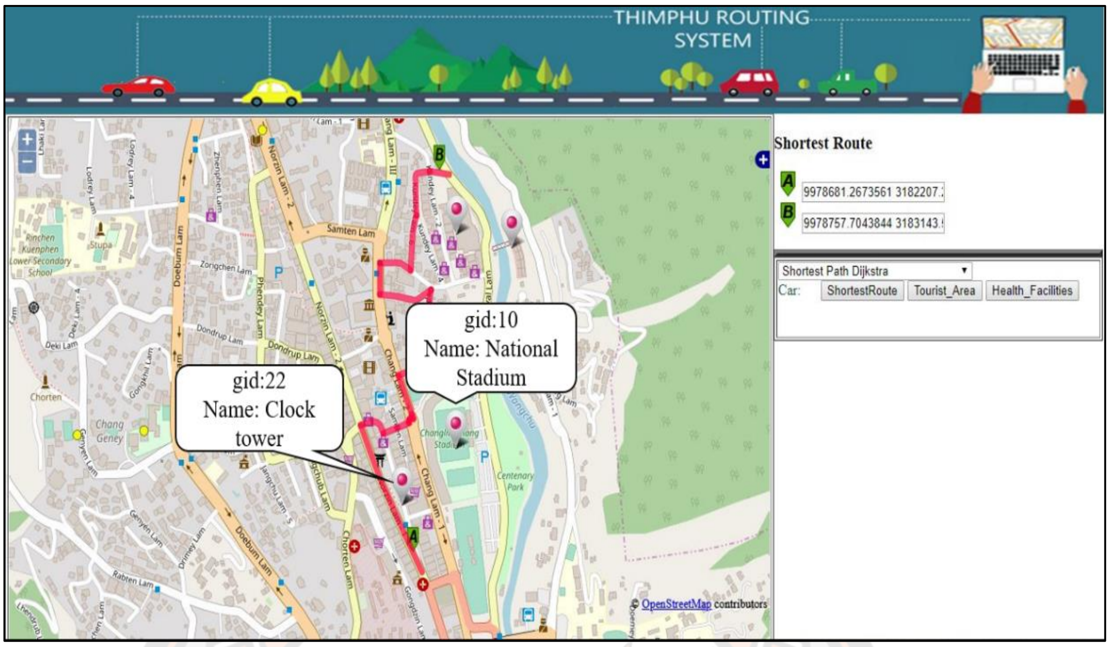


Figure 25 Web application showing the POI (Tourist Area) along the shortest route A to B

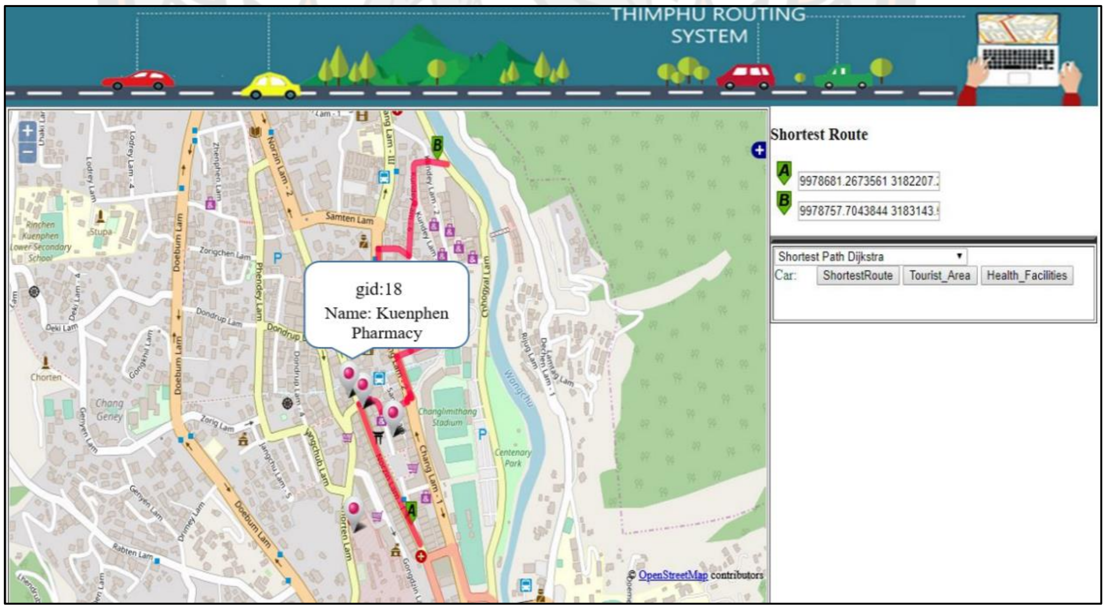
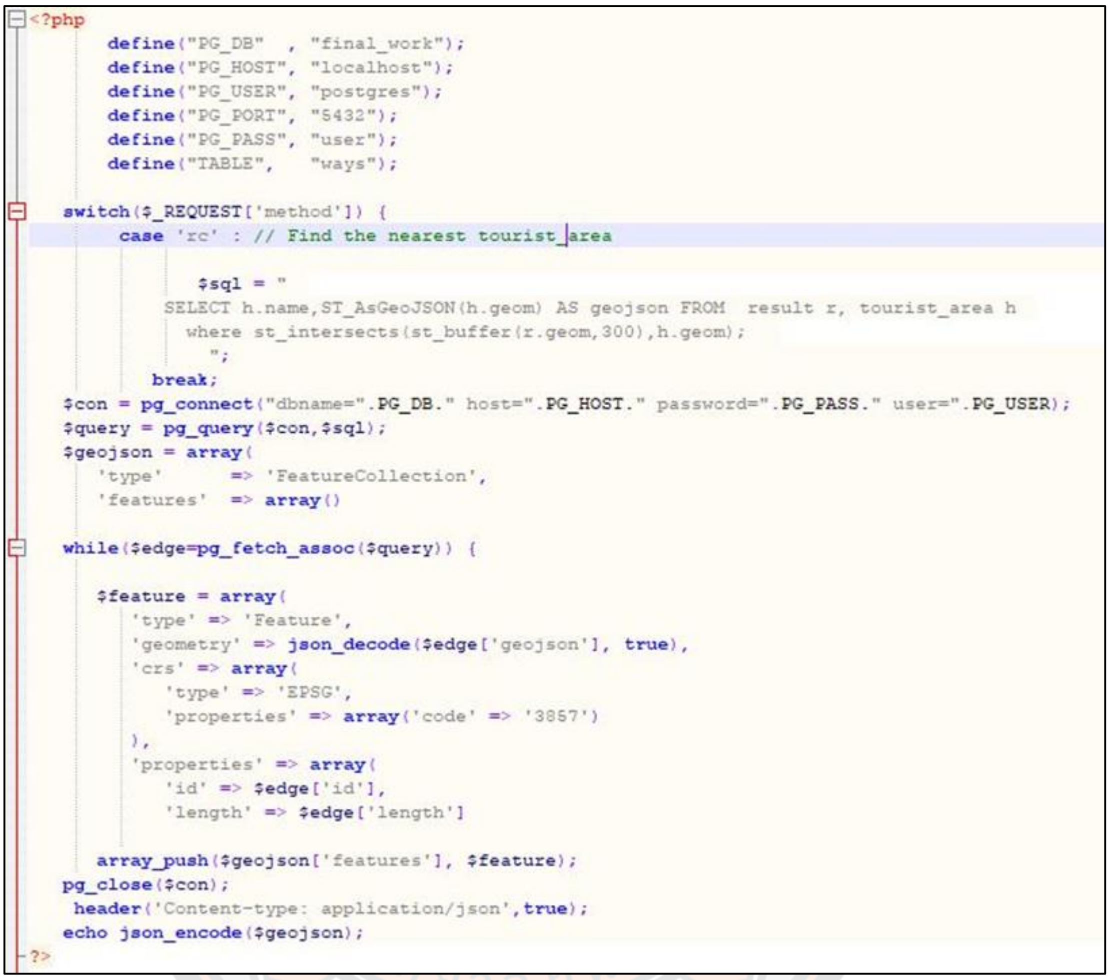


Figure 26 Web application showing the POI (Health_Facilities) along the shortest route A to B

The following is the code use to find the point of Interest for the tourist interested area: -



```php
<?php
        define("PG_DB"   , "final_work");
        define("PG_HOST", "localhost");
        define("PG_USER", "postgres");
        define("PG_PORT", "5432");
        define("PG_PASS", "user");
        define("TABLE",    "ways");

    switch($_REQUEST['method']) {
          case 'rc' : // Find the nearest tourist_area

               $sql = "
             SELECT h.name,ST_AsGeoJSON(h.geom) AS geojson FROM  result r, tourist_area h
               where st_intersects(st_buffer(r.geom,300),h.geom);
                 ";
            break;
    $con = pg_connect("dbname=".PG_DB." host=".PG_HOST." password=".PG_PASS." user=".PG_USER);
    $query = pg_query($con,$sql);
    $geojson = array(
       'type'       => 'FeatureCollection',
       'features'  => array()

    while($edge=pg_fetch_assoc($query)) {

       $feature = array(
          'type' => 'Feature',
          'geometry' => json_decode($edge['geojson'], true),
          'crs' => array(
             'type' => 'EPSG',
             'properties' => array('code' => '3857')
          ),
          'properties' => array(
             'id' => $edge['id'],
             'length' => $edge['length']

       array_push($geojson['features'], $feature);
    pg_close($con);
     header('Content-type: application/json',true);
    echo json_encode($geojson);
?>
```

4.1.2 Administrator Web Interface

The entire administration web interface was designed by using PHP, HTML and leaflet. The administrator-side interface was designed to update the cost values for the road network. The whole system/application is designed to show the routing result and to edit the value for the cost, so that the web can show the dynamic changes in the web.

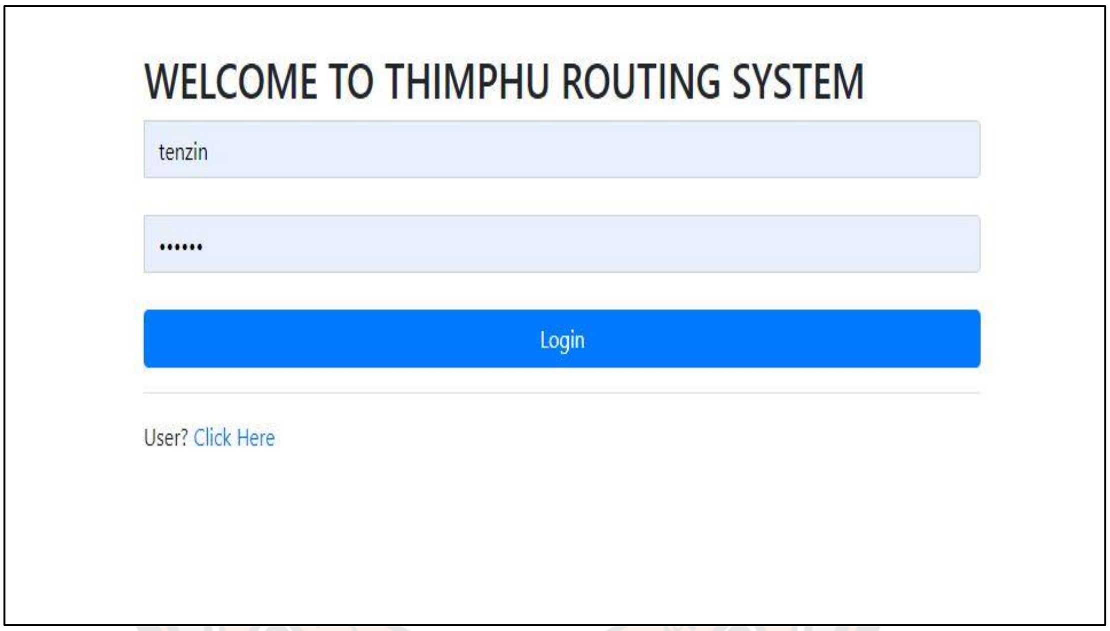
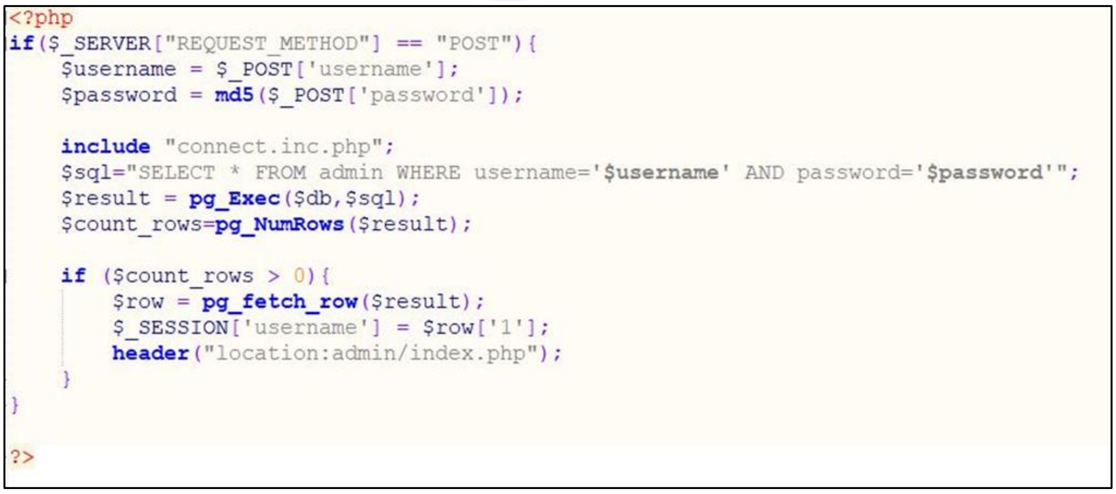


Figure 27 Showing the login page for the client/system administrator

The following is the php code for the login page: -

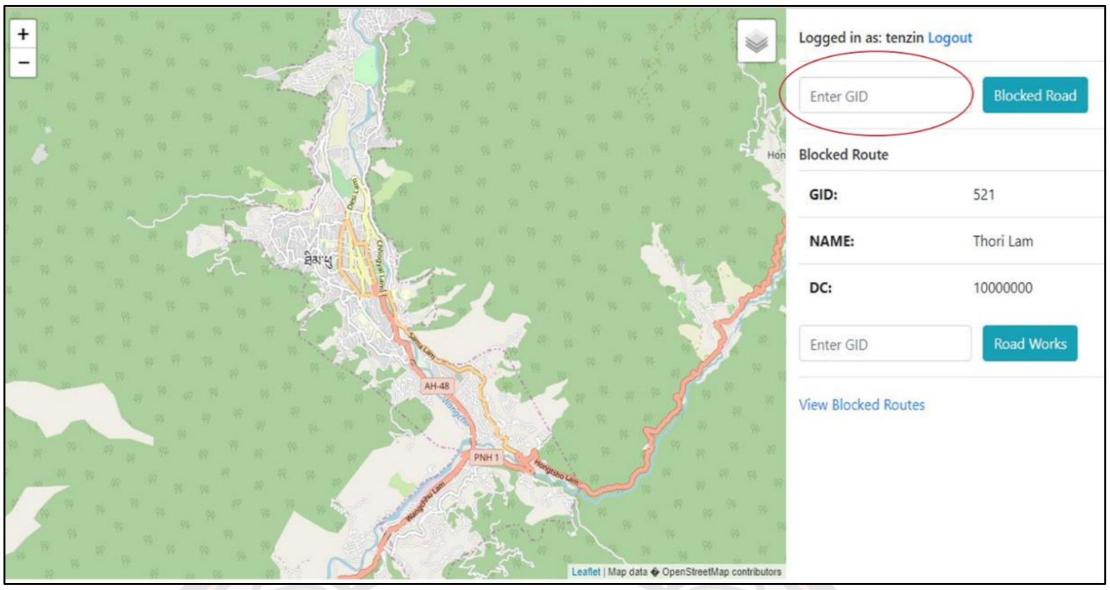


```
<?php
if($_SERVER["REQUEST_METHOD"] == "POST"){
    $username = $_POST['username'];
    $password = md5($_POST['password']);

    include "connect.inc.php";
    $sql="SELECT * FROM admin WHERE username='$username' AND password='$password'";
    $result = pg_Exec($db,$sql);
    $count_rows=pg_NumRows($result);

    if ($count_rows > 0){
        $row = pg_fetch_row($result);
        $_SESSION['username'] = $row['1'];
        header("location:admin/index.php");
    }
}

?>
```



Logged in as: tenzin Logout

Enter GID | Blocked Road

Blocked Route

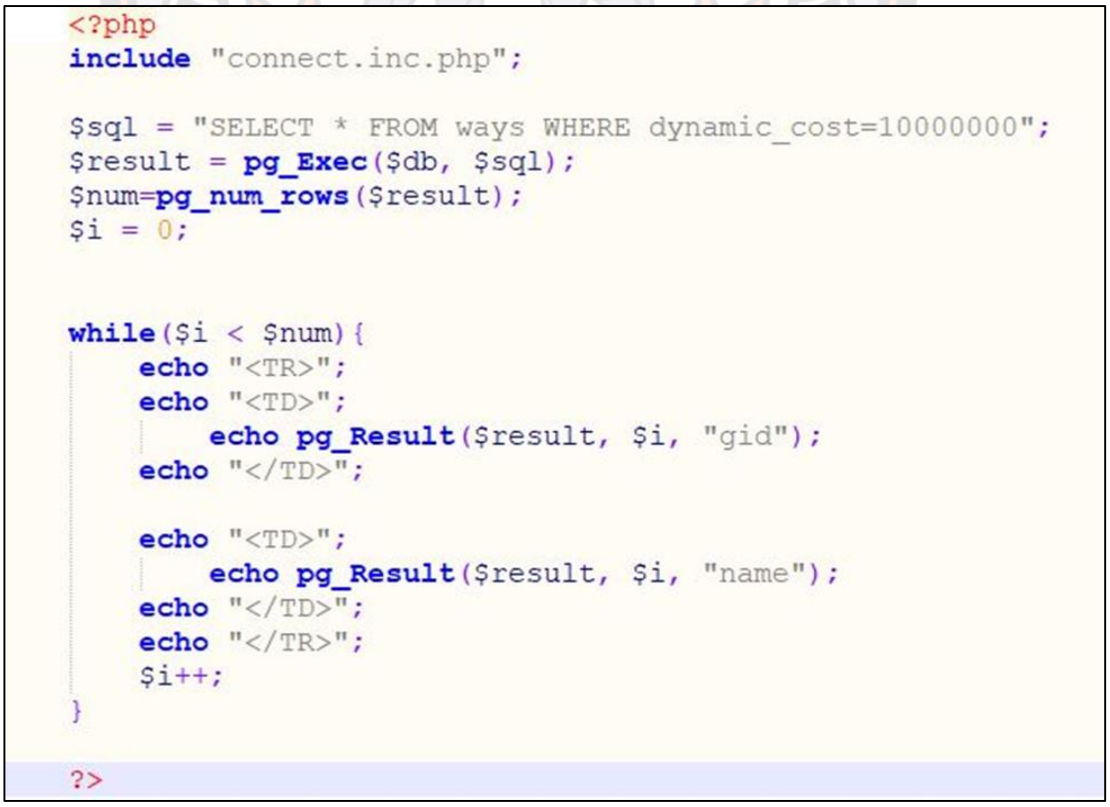
GID:	521
NAME:	Thori Lam
DC:	10000000

Enter GID | Road Works

View Blocked Routes

Figure 28 Web interface showing the how to update the Road Block.

The following is the php code for the system administrator to block the road by assigning the value for the dynamic cost as infinity: -



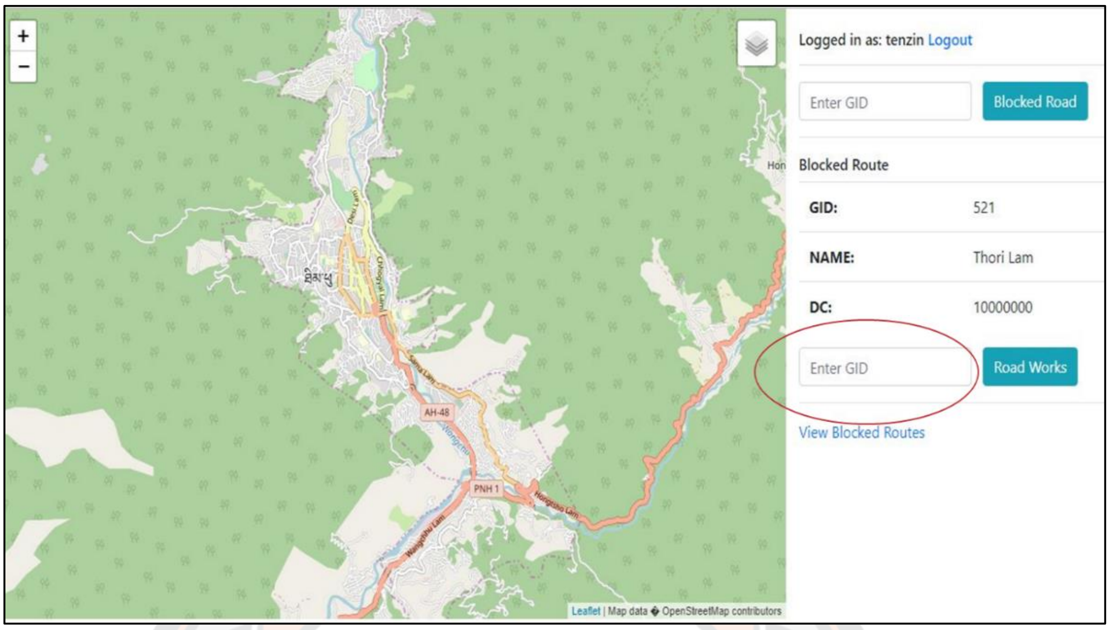
```
<?php
include "connect.inc.php";

$sql = "SELECT * FROM ways WHERE dynamic_cost=10000000";
$result = pg_Exec($db, $sql);
$num=pg_num_rows($result);
$i = 0;


while($i < $num){
    echo "<TR>";
    echo "<TD>";
        echo pg_Result($result, $i, "gid");
    echo "</TD>";

    echo "<TD>";
        echo pg_Result($result, $i, "name");
    echo "</TD>";
    echo "</TR>";
    $i++;
}

?>
```


Logged in as: tenzin Logout

Enter GID | Blocked Road

Blocked Route

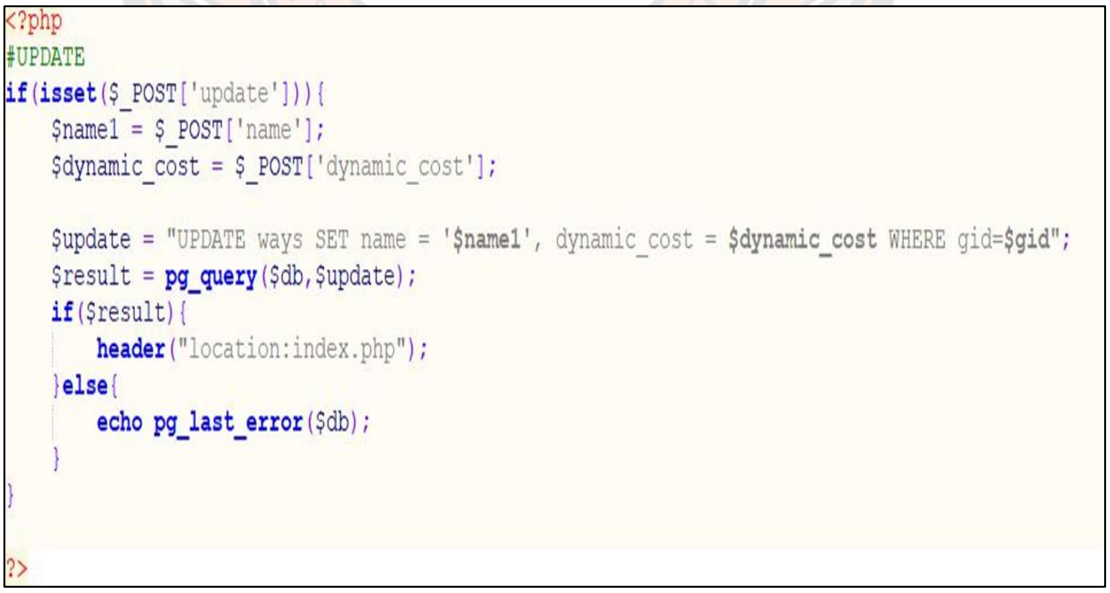
GID:	521
NAME:	Thori Lam
DC:	10000000

Enter GID | Road Works

View Blocked Routes

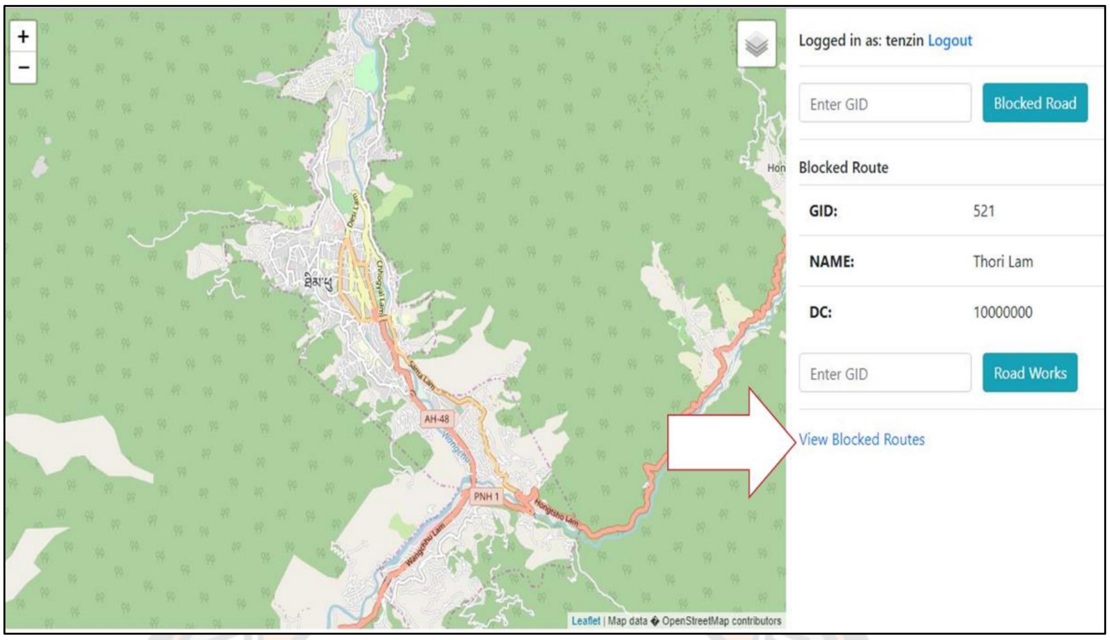
Figure 29 Web interface showing the how to update the Road when the temporary blocked is cleared.

The following is the php code for the system administrator to update the road information from the traffic police officer: -


```
<?php
#UPDATE
if(isset($_POST['update'])){
    $name1 = $_POST['name'];
    $dynamic_cost = $_POST['dynamic_cost'];

    $update = "UPDATE ways SET name = '$name1', dynamic_cost = $dynamic_cost WHERE gid=$gid";
    $result = pg_query($db,$update);
    if($result){
        header("location:index.php");
    }else{
        echo pg_last_error($db);
    }
}

?>
```

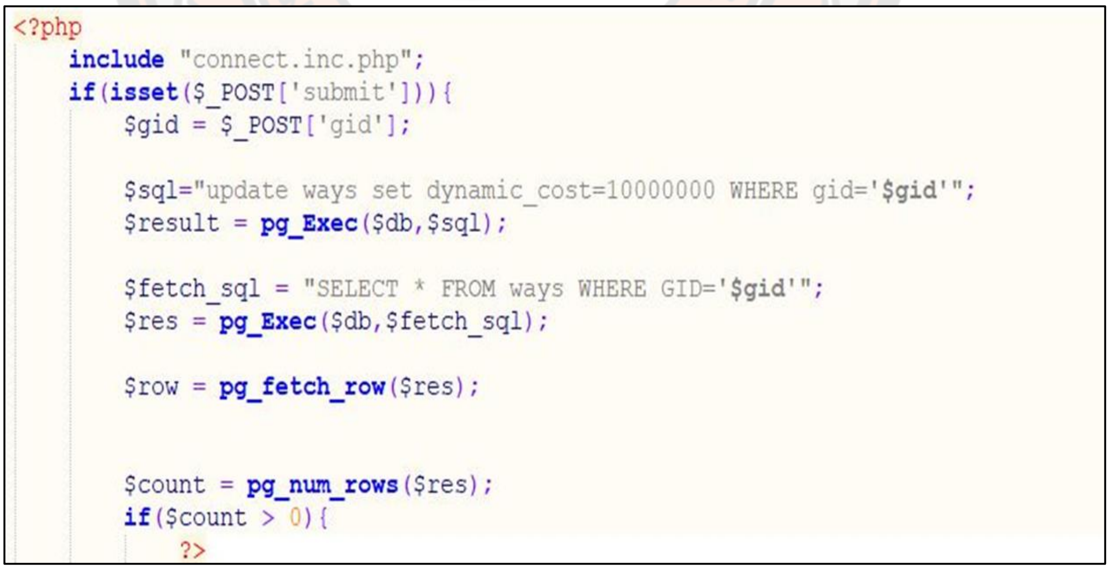

Logged in as: tenzin Logout
Enter GID
Blocked Road
Enter GID
Road Works
View Blocked Routes
Leaflet | Map data © OpenStreetMap contributors

Blocked Route

GID:	521
NAME:	Thori Lam
DC:	10000000

Figure 30 Web interface showing the how to view the detail of the road block.

If a system administrator wants to check the number of road being blocked due to traffic congestion or road accident, the following is the php code to show the number of road blocks.



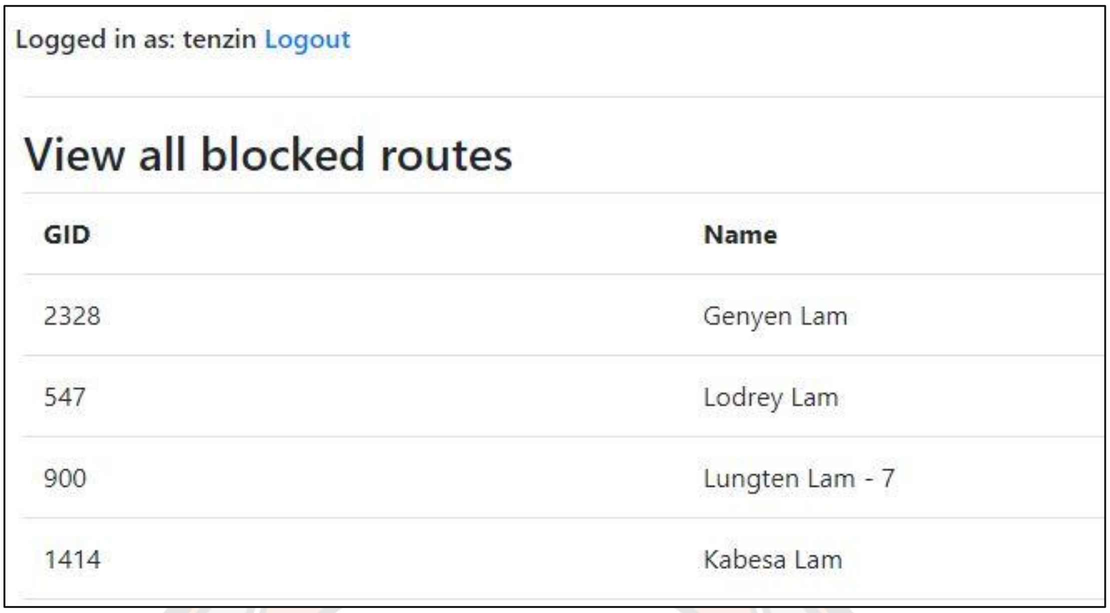
```
<?php
    include "connect.inc.php";
    if(isset($_POST['submit'])){
        $gid = $_POST['gid'];

        $sql="update ways set dynamic_cost=10000000 WHERE gid='$gid'";
        $result = pg_Exec($db,$sql);

        $fetch_sql = "SELECT * FROM ways WHERE GID='$gid'";
        $res = pg_Exec($db,$fetch_sql);

        $row = pg_fetch_row($res);


        $count = pg_num_rows($res);
        if($count > 0){
            ?>
```


Logged in as: tenzin Logout

View all blocked routes

GID	Name
2328	Genyen Lam
547	Lodrey Lam
900	Lungten Lam - 7
1414	Kabesa Lam

Figure 31 Web interface showing the details of the road blocks

CHAPTER V

DISCUSSION AND CONCLUSION

5.1 Discussion

The study developed a dynamic routing system for tourism based by using FOSS4G and modifying the pgRouting algorithm. pgRouting algorithm,such as Dijkstra algorithm, were enhanced, by modifying by taking into account dynamic changes in the road condition. The dynamic cost changes will act directly on the edge of increasing them in real time of an amount defined in the dynamic cost column. An alternative way is by use of True/False Boolean value by setting whether the road is blocked or not. The normal roads condition value can be set to False value, but incase if the road is blocked the value can be changed to true. If the value is False, the cost is 0 and if the value is true then the cost becomes infinity.

By developing the web interface for the routing service, it allows the clients to update the data through internet. The routing function can be created by using the PostGIS function to increase and decrease the cost value where the client/system administrator need not have to change it manually. pgRouting was not only used to modify the code for the length but also for the road conditions which will lead to the dynamic routing result. As compared to the other API in ArcGIS,google map and Open Street Map, pgRouting is has more flexibility due to the following reasons: -

- Data and attributes can be easily modified by many clients, like QGIS through JDBC, ODBC or directly using PI/pgSQL. The can also be PCs or mobile devices.
- Data changes can be reflected instantaneously through the routing engine and need not have to precalculated.
- The COST parameter can be dynamically calculated through SQL and its value can come from multiple fields or table.

Therefore, for this study FOSS4G and pgRouting was used to develop a routing system.

Especially if we are available to provide the traffic routing information, it will be useful for the driver and tourist to plan which route to take if there is a road under maintance or road accident. Hence, getting a real time information about the road situation is very useful for dynamic routing system. The system, if adopted by the Thimphu Municipality will ensure that there is a real time information updates and thus there will be efficient use of road network of the city.

5.2 Conclusion

Dijkstra algorithm can be used to modify the road condition for a dynamic change. This study suggests a new method of finding the concept of calculating the shortest path and supports dynamic changes of the road condition by using pgRouting functions. The weight of the Dynamic Cost will change as per the road condition. The weight will increase if there is a temporary blocked and will decrease when the road is being cleared.

For implementation of a system Open Source Software was used because the main reason was due to the strong community and enhancement of the software. Even a part of the software can be used by other developer to continue with developing of the algorithm.

This system can be used for the tourism management, traffic regulation and during the time of disasters/accidents.

Moreover, this system has a new feature showing the POI (Point of Interest) management functionality which can be stored in the DBMS. This system can show the POI for the health Facilities and tourist area along the shortest route which will be useful for the user.

5.3 Future work

Further work will be focused more towards development of Web-Based application on mobile devices and tablet PCs to improve the accessibility and usability for the user as well as the administrator. The updating of the road information dynamically through a mobile application will be very useful and easy to update the road information by the system administrator/traffic officer.

If the system is successful, it can be expanded to other major cities like Phuntsholing, Gelephu and Samdrup Jongkhar. These cities are also major receiver of migration and thus will see increasing population over the years.

Bhutan is vulnerable to a range of natural disasters such as earthquakes, Glacial Lake Outburst Flood (GLOF), flash floods, windstorms, forest fires and landslides. These disasters resulted into significant loss and damages to lives, properties and public infrastructures.

As per the Bhutan Disaster Risk Management Status Review, the two major earthquake in 2011 caused damage to around 7000 houses and many religious and public infrastructures. It was estimated that the earthquake affected around 7 percent of the population. Similarly, 1994 GLOF claimed 22 lives and damaged many infrastructure and properties. The risk of GLOF disaster is very huge as there are 77 glaciers and 2674 glacial lakes with potential risk. In 2009, the Cyclone Aila floods caused a loss of more than Nu. 700 million.

However, like most of the countries across the globe, Bhutan lacks information and data about the risk assessment of disaster. Therefore, the system with little modification can be very helpful in mitigating and adapting to disaster risk.

Appendix 1

This Appendix is the main PHP,JavaScript and HTML code for the web application.

```
<html>
<head>
<title>Thimphu Routing System</title>
<div align=”center”><img src="banner.png" width="100%" height="110" ></div>
<script type="text/javascript" src="http://media.scraperwiki.com/js/jquery-
1.5.2.min.js"></script>
<script type="text/javascript"
src="http://www.openlayers.org/api/OpenLayers.js"></script>
<script type="text/javascript"
src="http://www.openstreetmap.org/openlayers/OpenStreetMap.js"></script>
<script src="http://maps.google.com/maps/api/js?v=3.6&amp;sensor=false"></script>
<script type="text/javascript">
var  start, stop, result, ems, controls;
var map,layer, click, status;


OpenLayers.Control.Click = OpenLayers.Class(OpenLayers.Control, {
   defaultHandlerOptions: {
          'single': true,
          'double': false,
          'pixelTolerance': 0,
          'stopSingle': false,
          'stopDouble': false
   },

   initialize: function(options) {
          this.handlerOptions = OpenLayers.Util.extend(
             {}, this.defaultHandlerOptions
          );
```

```
        OpenLayers.Control.prototype.initialize.apply(
            this, arguments
        );
        this.handler = new OpenLayers.Handler.Click(
            this, {
                    'click': this.trigger
            }, this.handlerOptions
        );
    },
    trigger: function(e) {

    //var toProjection = new OpenLayers.Projection("EPSG:4326"); //chai edit
   // var xy = map.getLonLatFromPixel(e.xy).transform(map.getProjectionObject(),
toProjection); //chai edit
   //alert (xy);

    var xy = map.getLonLatFromPixel(e.xy);
    var retOut = ""+xy.lon+" "+xy.lat+"";
    if (status == 'start') {
        $("#start").val(retOut);
        start.removeFeatures(start.features );
        //alert (retOut);

    var poi = new OpenLayers.Geometry.Point(xy.lon, xy.lat);


        //alert (poi);
        var feature = new OpenLayers.Feature.Vector(poi);
      start.addFeatures([feature]);
        simpleProcessing();
    } else if (status == 'end') {
        $("#end").val(retOut);
        stop.removeFeatures(stop.features );
```

```
            var poi = new OpenLayers.Geometry.Point(xy.lon, xy.lat);
            var feature = new OpenLayers.Feature.Vector(poi);
        stop.addFeatures([feature]);
        simpleProcessing();
    }
    }

});
function setcoordinates(val){
  click.activate();
  status = val;
}

OpenLayers.ProxyHost = "geoproxy.php?url="

            var map;
            var y=27.46735; //100.27792,16.81836
            var x=89.64445;
            var zoom=13;

                        //Initialise the 'map' object

            function init(){
                        //var option =
{controls,maxExtent:bounds,maxResolution:"auto",units:"m",
                        //
projection:prjGoogle,displayProjection:4326,eventListeners:{'moveend':alert_on_mov
e}};
            map = new OpenLayers.Map({
                        div: "map",
                        projection: "EPSG:3857",
                        displayProjection: "EPSG:4326",
            maxResolution: 'auto',
    });

            //map = new OpenLayers.Map('map_element',option);
            // Add Map Control
            //map.addControl(new OpenLayers.Control.Navigation());
    map.addControl(new OpenLayers.Control.Attribution());
    map.addControl(new OpenLayers.Control.MousePosition());
            //map.addControl(new OpenLayers.Control.PanZoomBar());
    map.addControl(new OpenLayers.Control.LayerSwitcher());

            // Add Base Map
    var mapnik = new OpenLayers.Layer.OSM();
    var gphy = new OpenLayers.Layer.Google(
            "Google Physical",
            (Chandniha et al., 2017)//add data
    );
    var gmap = new OpenLayers.Layer.Google(
            "Google Streets", // the default
            {type:google.maps.MapTypeId.ROADMAP, numZoomLevels:
20}//add data
    );
    var ghyb = new OpenLayers.Layer.Google(
            "Google Hybrid",
            {type:google.maps.MapTypeId.HYBRID, numZoomLevels: 20}//add
data
    );
    var gsat = new OpenLayers.Layer.Google(
            "Google Satellite",
            {type:google.maps.MapTypeId.SATELLITE, numZoomLevels:
22}//add data
    );
```

```
	var lonlat = new OpenLayers.LonLat(x, y).transform(
		new OpenLayers.Projection("EPSG:4326"), // transform from WGS
1984
		new OpenLayers.Projection("EPSG:900913") // to Spherical Mercator
	);

	//Add map layers
	map.addLayers([mapnik,gmap,gphy,ghyb,gsat]);

	map.setCenter(new OpenLayers.LonLat(x, y).transform(
		new OpenLayers.Projection("EPSG:4326"),
		map.getProjectionObject()
	), zoom);

	// Add GeoServer Layers
	var road = new
OpenLayers.Layer.WMS("Thimphu_Road","http://localhost:8088/geoserver/city/wms
?", {layers:'city:ways', transparent:true},{visibility:true});
	var Health = new
OpenLayers.Layer.WMS("Health_Facilities","http://localhost:8088/geoserver/city/w
ms?", {layers:'city:Health_Facilities', transparent:true},{visibility:true});
	var tourist = new
OpenLayers.Layer.WMS("Tourist_Area","http://localhost:8088/geoserver/city/wms?"
, {layers:'city:Tourist_area', transparent:true},{visibility:true});

	map.addLayers([road,Health,tourist]);

	// Add GeoJSON Layers
	var plk_road =  new OpenLayers.Layer.Vector('Road', {
      strategies: [new OpenLayers.Strategy.Fixed()],
      protocol: new OpenLayers.Protocol.HTTP({
	url: '/data/plk_roads.geojson',
          format: new OpenLayers.Format.GeoJSON()
        }),
        //style: {fillColor: '#382', strokeColor: 'yellow', strokeWidth: 1, strokeOpacity:
0.8},
        styleMap: new OpenLayers.StyleMap({
          "default": new OpenLayers.Style(null, {
            rules: [new OpenLayers.Rule({
              title: 'ถนน',
              symbolizer: {
                "Line": {
                  fillColor: '#1C1C1C',
                  fillOpacity: 0.3,
                  graphicName: "circle",
                  strokeColor: '#1C1C1C',
                              strokeOpacity: 0.5,
                  strokeWidth: 1.5,
                  graphicZIndex: 1,
                  pointRadius: 2
                }
              }
            })]
          })
        }),
        projection: new OpenLayers.Projection("EPSG:3857"),
        visibility: false
    });

var plk_hospitals =  new OpenLayers.Layer.Vector('Hospital', {
        strategies: [new OpenLayers.Strategy.Fixed()],
        protocol: new OpenLayers.Protocol.HTTP({
          url: 'data/plk_hospitals.geojson',
          format: new OpenLayers.Format.GeoJSON()
    }),
    //style: {fillColor: '#382', strokeColor: 'yellow', strokeWidth: 1, strokeOpacity:
0.8},
                styleMap: new OpenLayers.StyleMap({
                        "default": new OpenLayers.Style(null, {
                            rules: [new OpenLayers.Rule({
                              title: 'โรงพยาบาล',
                              symbolizer: {
                                  "Point": {
                                  externalGraphic: "img/hos.png",
                                  graphicWidth: 20,
                                  graphicHeight: 20,
                                  graphicYOffset: -26,
                                  graphicOpacity: 1
                                  }
                              }
                            })]
                        })
                    }),
    projection: new OpenLayers.Projection("EPSG:3857"),
    visibility: true
  });


      click = new OpenLayers.Control.Click();
      map.addControl(click);

  var start_style = OpenLayers.Util.applyDefaults({
      externalGraphic: "img/start.png",
      graphicWidth: 14,
      graphicHeight: 26,
      graphicYOffset: -26,
      graphicOpacity: 1
  }, OpenLayers.Feature.Vector.style['default']);

  var stop_style = OpenLayers.Util.applyDefaults({
      externalGraphic: "img/stop.png",
      graphicWidth: 14,
      graphicHeight: 26,
      graphicYOffset: -26,
      graphicOpacity: 1
  }, OpenLayers.Feature.Vector.style['default'])
  var ems_style = OpenLayers.Util.applyDefaults({
      externalGraphic: "img/pinkmark.png",
      graphicWidth: 40,
      graphicHeight: 40,
      graphicYOffset: -26,
      graphicOpacity: 1
  }, OpenLayers.Feature.Vector.style['default']);
  var result_style = OpenLayers.Util.applyDefaults({
       strokeWidth: 7,
       strokeColor: "#FF0040",
       hoverFillOpacity: 0.7,
       strokeOpacity: 0.7,
       fillOpacity: 0.6
  }, OpenLayers.Feature.Vector.style['default']);
      start   = new OpenLayers.Layer.Vector("Start point", {style: start_style});
      stop    = new OpenLayers.Layer.Vector("End point1", {style: stop_style});
      result  = new OpenLayers.Layer.Vector("First route", {style: result_style});
      tour    = new OpenLayers.Layer.Vector("Tourist POI", {style: ems_style});
      he      = new OpenLayers.Layer.Vector("he POI", {style: ems_style});

      map.addLayers([start,stop,result,tour,he]);
```

```
            //pgRouting start here (add start stop)
            }
function simpleProcessing() {
  var lon = $("#start").val();

  var lat = $("#end").val();
  var mtd = $("#method").val();

$.ajax({
  url:
url+'startpoint='+lon+'&finalpoint='+lat+'&method='+mtd+'&region=ways&srid=385
7',
  success: function(data){
      var GeoJSON = new OpenLayers.Format.GeoJSON();
      var features = GeoJSON.read(data);
      result.removeFeatures(result.features);
      result.addFeatures(features);
  }
});
 //alert (feature);
var controls = {
                        start:  new OpenLayers.Control.DrawFeature(start),
                        stop:   new OpenLayers.Control.DrawFeature(stop)
                        //go:   new OpenLayers.Control.DrawFeature(go)
             }

             for (var key in controls) {
                        map.addControl(controls[key]);
             }
            }
function touristpoi() {
  var url = 'http://localhost/example_routing_on_web/work/find_poi.php?';
  var nearh = $("#start").val();

  var lat = $("#end").val();
// if (isNaN(lat)) {return alert("End point not found!")};

//url+='startpoint='+lon+'&finalpoint='+lat+'&method=rc&region=plk_roads&srid=38
57';
  url+='startpoint='+nearh+'&method=rc&region=ways&srid=3857';

//alert (url);
//alert (lon);
//alert (lat);
//alert (features);

//var urlroute = 'http://localhost/routing/work/routing_nu_new.php?';
//var poihos =

$.ajax({
  url: url,
  success: function(data){
      var GeoJSON = new OpenLayers.Format.GeoJSON();
      var features = GeoJSON.read(data);
      tour.removeFeatures(tour.features);
      tour.addFeatures(features);
  }

});
var controls = {
                         ems:  new OpenLayers.Control.DrawFeature(ems)


              }
```

```
            for (var key in controls) {
                    map.addControl(controls[key]);
             }
           }
function healthpoi() {

  var url = 'http://localhost/example_routing_on_web/work/find_poi_health.php?';
  var nearh = $("#start").val();

  var lat = $("#end").val();
// if (isNaN(lat)) {return alert("End point not found!")};

//url+='startpoint='+lon+'&finalpoint='+lat+'&method=rc&region=plk_roads&srid=38
57';
  url+='startpoint='+nearh+'&method=rc&region=ways&srid=3857';

//alert (url);
//alert (lon);
//alert (lat);
//alert (features);

//var urlroute = 'http://localhost/routing/work/routing_nu_new.php?';
//var poihos =

$.ajax({
  url: url,
  success: function(data){
     var GeoJSON = new OpenLayers.Format.GeoJSON();
     var features = GeoJSON.read(data);
     he.removeFeatures(he.features);
     he.addFeatures(features);
  }


});
var controls = {
                         ems:  new OpenLayers.Control.DrawFeature(ems)


              }


              for (var key in controls) {
                         map.addControl(controls[key]);
              }
            }
function toggleControl(element) {
            for (key in controls) {
                                              if (element.value == key && element.checked) {
                    controls[key].activate();
                  }     else {
                    controls[key].deactivate();
                                                       }
            }


                       //alert(element.value);
    }
</script>
</head>
<body onload="init()">
   <table width="70%" height="90%" align="left" border="1" bdcolor="red">
            <tr> <td>
            <div id="map" style="width: 100%; height: 100%;"></div>
            </td></tr>
  </table>
<table width="30%" height="100%" align="left" border="4">
```

```
    <h3> Shortest Route </h3>
          <ul>
              &nbsp; <img src="start.png" alt="Start point" width="18" height="26"
/>&nbsp;
              <input id="start" onclick="setcoordinates('start');" />
              <br>
              &nbsp; <img src="stop.png" alt="End point" width="18" height="26"
/>&nbsp;
              <input id="end" onclick="setcoordinates('end');" />
              <br>
           </ul>
</table>
<!--
<table   width="30%" border="1" bdcolor="red"><tr> <td>

 <input type="radio" name="type" value="select" id="selectToggle"
        onclick="toggleControls(this);" />
  <label for="selectToggle"><font color=#156854 bgcolor="black">Road closed
information</font></label>

</td></tr></table> -->
<table   width="30%" border="1" bdcolor="red"><tr> <td>
   <select id="method">
    <option value=>...Choose your option...</option>
    <option value="ST">Shortest Times </option>
    <option value="SPDD">Shortest Path Dijkstra</option>
    <option value="RTD">Main Roads and Safest</option>
     <option value="SPSB">Shortest Path Shooting Star -</option>
   <!--
    <option value="SPDD">Shortest Path Dijkstra - directed (BBox)</option>
    <option value="SPA">Shortest Path A Star – spdd directed</option>
    <option value="SPSB">Shortest Path Shooting Star - </option>
    <option value="SPS">Shortest Path Shooting Star</option>
   </select><br>
   <label ><font color=#156854 bgcolor="black">Car:&nbsp&nbsp
&nbsp&nbsp</font></label>
   <input type="button" value="ShortestRoute"
  <input type="button" onclick="touristpoi();" value="Tourist_Area" />
  <input type="button" onclick="healthpoi();" value="Health_Facilities" />
</table><br><br><br><br><br><br><br><br><br>
  <!--<div id="AB" align = "lefe">&nbsp; </div>-->
  <div id="CD" align = "lefe">&nbsp; </div>   <!-- route 1 table-->
  <div id="CD2" align = "lefe">&nbsp; </div>  <!-- route 2 table-->
 <!-- <div id="CD1" align = "lefe">&nbsp; </div> --> <!-- reverse table-->
  </body>
</html>
```

Appendix 2

Server side PHP script

This appendix script of PHP makes the routing query from starting/ending coordinates by clicking on the map. It searches for the nearest edge to get the staring node and ending node to call the pgRouting function and send the result to the web interface.

```
          define("PG_DB"  , "final_work");
          define("PG_HOST", "localhost");
          define("PG_USER", "postgres");
          define("PG_PORT", "5432");
          define("PG_PASS", "user");
          define("TABLE",   "ways");

  // Retrieve start point
 // $start='100.19533576766561 16.752196023246402';
  $start = explode(' ',$_REQUEST['startpoint']);
  $startPoint = array($start[0], $start[1]);

  // Retrieve end point
 // $end = '100.1969665507453 16.73600428680829';
  $end = explode(' ',$_REQUEST['finalpoint']);
  $endPoint = array($end[0], $end[1]);
  //echo $startPoint;
  // Find the nearest edge
  $startEdge = findNearestEdge($startPoint);
  $endEdge   = findNearestEdge($endPoint);
  // FUNCTION findNearestEdge
  function findNearestEdge($lonlat) {

    // Connect to database
    $con = pg_connect("dbname=".PG_DB." host=".PG_HOST."
password=".PG_PASS." user=".PG_USER);
```

```
    $sql = "SELECT gid, source, target, geom,
                     ST_Distance(geom, ST_GeometryFromText(
                   'POINT(".$lonlat[0]." ".$lonlat[1].")', 3857)) AS dist
               FROM ".TABLE."

               ORDER BY dist LIMIT 1";

                              // echo "<br>";
                              // echo $sql;
    $query = pg_query($con,$sql);

    $edge['gid']      = pg_fetch_result($query, 0, 0);
    $edge['source']   = pg_fetch_result($query, 0, 1);
    $edge['target']   = pg_fetch_result($query, 0, 2);
    $edge['geom'] = pg_fetch_result($query, 0, 3);

    // Close database connection
    pg_close($con);

/*
echo $sql;
echo "<br>";
echo $edge['gid'];
echo "<br>";
echo $edge['source'];
echo "<br>";
echo $edge['target'];
echo "<br>";
echo $edge['the_geom'];
echo "<br>";
echo "<br>;

    return $edge;

  }
  // Select the routing algorithm
  switch($_REQUEST['method']) {

                case 'SPDD' : // Shortest Path Dijkstra*

                  $sql =
                        drop table if exists result;
                        create table result (gid bigint);
                        select
addgeometrycolumn('result','geom',3857,'MULTILINESTRING',2);
                        insert into result(gid,geom)
                        SELECT gid, geom FROM ways w,
                  (SELECT * FROM pgr_dijkstra(
                        'SELECT gid as id,
                        source,
                        target,
                        length + dynamic_cost AS cost
                        FROM ways',
                        ".$startEdge['source'].",
                                ".$endEdge['target'].",
                        directed := false)

                        )as rt
                        where w.gid=rt.edge;

                        SELECT ST_AsGeoJSON(w.geom) AS geojson, geom FROM
ways w,
                  (SELECT * FROM pgr_dijkstra(
                        'SELECT gid as id,
                        source,
                        target,
                        length + dynamic_cost AS cost
                        FROM ways',
                        ".$startEdge['source'].",
                                ".$endEdge['target'].",
                        directed := false)

                        )as rt
                        where w.gid=rt.edge;

                        ";

                  break;

              $sql = "
                        drop table if exists result;
                        create table result (gid bigint);
                        select
addgeometrycolumn('result','geom',3857,'MULTILINESTRING',2);
                        insert into result(gid,geom)
                        SELECT gid, geom FROM ways w,
                  (SELECT * FROM pgr_dijkstra(
                        'SELECT gid as id,
                        source,
                        target,
                        length + dynamic_cost AS cost
                        FROM ways',
                        ".$startEdge['source'].",
                                ".$endEdge['target'].",
                              directed := false)

                              )as rt
                              where w.gid=rt.edge;

                              SELECT h.name,ST_AsGeoJSON(h.geom) AS geojson FROM
result r, poi h
                                where st_intersects(st_buffer(r.geom,1000),h.geom);

                         ";
                        // echo $sql;

                           break;



                      } // close switch
// Connect to database
  $con = pg_connect("dbname=".PG_DB." host=".PG_HOST."
password=".PG_PASS." user=".PG_USER);

  // Perform database query
  $query = pg_query($con,$sql);

  //echo $sql;

          // Return route as GeoJSON
  $geojson = array(
    'type'      => 'FeatureCollection',
    'features'  => array()
  );

  // Add edges to GeoJSON array
```

```
while($edge=pg_fetch_assoc($query)) {

  $feature = array(
    'type' => 'Feature',
    'geometry' => json_decode($edge['geojson'], true),
    'crs' => array(
      'type' => 'EPSG',
      'properties' => array('code' => '3857')
    ),
    'properties' => array(
      'id' => $edge['id'],
      'length' => $edge['length']
    )
  );

  // Add feature array to feature collection array
  array_push($geojson['features'], $feature);
}


// Close database connection
pg_close($con);

// Return routing result
 header('Content-type: application/json',true);
echo json_encode($geojson);
```

Appendix 3

PHP code for finding the Point of Interest(POI) for Health Facilities.

```
// Database connection settings
            define("PG_DB"  , "final_work");
            define("PG_HOST", "localhost");
            define("PG_USER", "postgres");
            define("PG_PORT", "5432");
            define("PG_PASS", "user");
            define("TABLE",   "ways");
  // Retrieve start point
  // Select the routing algorithm
  switch($_REQUEST['method']) {
                   case 'rc' : // Find the nearest hospital

                  $sql = "
                            SELECT h.name,ST_AsGeoJSON(h.geom) AS geojson FROM
result r, health_facilities h
                              where st_intersects(st_buffer(r.geom,500),h.geom)
                   ";
                  // echo $sql;

                     break;
               } // close switch

// Connect to database
  $con = pg_connect("dbname=".PG_DB." host=".PG_HOST."
password=".PG_PASS." user=".PG_USER);

  // Perform database query
  $query = pg_query($con,$sql);

  //echo $sql;
```

```
        // Return route as GeoJSON
$geojson = array(
  'type'      => 'FeatureCollection',
  'features'  => array()
);

// Add edges to GeoJSON array
while($edge=pg_fetch_assoc($query)) {

  $feature = array(
    'type' => 'Feature',
    'geometry' => json_decode($edge['geojson'], true),
    'crs' => array(
      'type' => 'EPSG',
      'properties' => array('code' => '3857')
    ),
    'properties' => array(
      'id' => $edge['id'],
      'length' => $edge['length']
    )
  );

  // Add feature array to feature collection array
  array_push($geojson['features'], $feature);
}
// Close database connection
pg_close($con);

// Return routing result
 header('Content-type: application/json',true);
echo json_encode($geojson)
```

Appendix 4

PHP code for finding the Point of Interest(POI) for Tourist Area.

```
<?php
  // Database connection settings
            define("PG_DB"  , "final_work");
            define("PG_HOST", "localhost");
            define("PG_USER", "postgres");
            define("PG_PORT", "5432");
            define("PG_PASS", "user");
            define("TABLE",   "ways");

  // Retrieve start point
  // Select the routing algorithm
  switch($_REQUEST['method']) {
                  case 'rc' : // Find the nearest tourist_area

                 $sql = "
                           SELECT h.name,ST_AsGeoJSON(h.geom) AS geojson FROM
result r, tourist_area h
                            where st_intersects(st_buffer(r.geom,300),h.geom);
                  ";
                 // echo $sql;

                    break;

               } // close switch
// Connect to database
  $con = pg_connect("dbname=".PG_DB." host=".PG_HOST."
password=".PG_PASS." user=".PG_USER);

  // Perform database query
```

```
$query = pg_query($con,$sql);

//echo $sql;

        // Return route as GeoJSON
$geojson = array(
  'type'      => 'FeatureCollection',
  'features'  => array()
);

// Add edges to GeoJSON array
while($edge=pg_fetch_assoc($query)) {

  $feature = array(
    'type' => 'Feature',
    'geometry' => json_decode($edge['geojson'], true),
    'crs' => array(
      'type' => 'EPSG',
      'properties' => array('code' => '3857')
    ),
    'properties' => array(
      'id' => $edge['id'],
      'length' => $edge['length']
    )
  );

  // Add feature array to feature collection array
  array_push($geojson['features'], $feature);
}
// Close database connection
pg_close($con);
```

```
  // Return routing result
   header('Content-type: application/json',true);
  echo json_encode($geojson);
?>
```

Appendix 5

This Appendix from 5 to 9 are the main PHP and HTML code for the web application for the administrator side.

```
<?php session_start(); ?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="http://localhost/example_routing_on_web/work/bootstrap/css/bootstrap.min.css
">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script
src="http://localhost/example_routing_on_web/work/bootstrap/js/bootstrap.min.js"><
/script>
</head>
<body>
<div class="container" style="margin-top: 30px">
<div class="row">
<div class="col-sm-2"></div>

<div class="col-sm-8">
<h2>WELCOME TO THIMPHU ROUTING SYSTEM</h2>
<div class="panel panel-default">
<div class="panel-body">
<form id="login-form" method="POST" role="form">
<?php if (isset($_GET['err'])) { ?>
<div class="alert alert-danger text-center"><?php echo "Login failed! Invalid email-id
or password!"; ?></div>
<?php } ?>
<div class="input-group">
<span class="input-group-addon"><i class="glyphicon glyphicon-
envelope"></i></span>
<input type="text" name="username" placeholder="Username" required class="form-
control" />
</div>
<br>
<div class="input-group">
<span class="input-group-addon"><i class="glyphicon glyphicon-lock"></i></span>
<input type="password" name="password" placeholder="Password" required
class="form-control" />
</div>
```

```
<br>
<div class="form-group">
<input type="submit" name="submit" value="Login" class="btn btn-primary btn-
block" />
</div>

<div class="form-group">
<hr/>
<div class="col-sm-6" style="padding: 0;">User? <a
href="http://localhost/example_routing_on_web/work/work6_pgrouting.html">Click
Here</a></div>
</div>
</form>
</div>
</div>
</div>

<div class="col-sm-2"></div>
</div>
</div>
</body>
</html>

<?php
if($_SERVER["REQUEST_METHOD"] == "POST"){
$username = $_POST['username'];
$password = md5($_POST['password']);

include "connect.inc.php";
$sql="SELECT * FROM admin WHERE username='$username' AND
password='$password'";
$result = pg_Exec($db,$sql);
$count_rows=pg_NumRows($result);

if ($count_rows > 0){
$row = pg_fetch_row($result);
$_SESSION['username'] = $row['1'];
header("location:admin/index.php");
}
}
```

Appendix 6

```
<?php
session_start();

if(!isset($_SESSION['username'])){
header("location:../index.php");
exit();
}
?>

<!DOCTYPE html>
<html>
<head>
<title>Leaflet Map</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="shortcut icon" type="image/x-icon" href="docs/images/favicon.ico" />
<link rel="stylesheet"
href="http://localhost/example_routing_on_web/work/bootstrap/css/bootstrap.min.css
">
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.5.1/dist/leaflet.css"
integrity="sha512-
xwE/Az9zrjBIphAcBb3F6JVqxf46+CDLwfLMHloNu6KEQCAWi6HcDUbeOfBIpt
F7tcCzusKFjFw2yuvEpDL9wQ==" crossorigin=""/>
<script src="https://unpkg.com/leaflet@1.5.1/dist/leaflet.js" integrity="sha512-
GffPMF3RvMeYyc1LWMHtK8EbPv0iNZ8/oTtHPx9/cc2ILxQ+u905qIwdpULaqDk
yBKgOaB57QTMg7ztg8Jm2Og==" crossorigin=""></script>
<script
src="http://localhost/example_routing_on_web/work/bootstrap/js/bootstrap.min.js"><
/script>
</head>
<body>
<div class="container-fluid">
<div class="row">
<div class="col-sm-8" id="mapid" style="width: 100%; height: 600px;"></div>

<div class="col-sm-4" style="margin-top: 20px;">
<h6>Logged in as: <?php echo $_SESSION['username']; ?> <a
href="http://localhost/example_routing_on_web/admin/logout.php">Logout</a></h6
>

<hr>

<form class="form-inline" method="POST">
<div lass="form-group">
<input type="text" name="gid" placeholder="Enter GID" class="form-
control"></input>
```

```
</div>
&nbsp;&nbsp;&nbsp;
<button type="submit" name = "submit" class="btn btn-info">Blocked
Road</button>
</form>
<hr>

<?php
include "connect.inc.php";
if(isset($_POST['submit'])){
$gid = $_POST['gid'];

$sql="update ways set dynamic_cost=10000000 WHERE gid='$gid'";
$result = pg_Exec($db,$sql);

$fetch_sql = "SELECT * FROM ways WHERE GID='$gid'";
$res = pg_Exec($db,$fetch_sql);

$row = pg_fetch_row($res);


$count = pg_num_rows($res);
if($count > 0){
?>
<h6>Blocked Route</h6>
<table class="table">
<tr>
<th>GID:</th>
<td><?php echo $row[0] ?></td>
</tr>

<tr>
<th>NAME:</th>
<td><?php echo $row[6] ?></td>
</tr>

<tr>
<th>Dynamic Cost:</th>
<td><?php echo $row[25] ?></td>
</tr>
</table>
<?
}else{
?>
<div class="alert alert-warning" role="alert">
GID <?php echo $gid; ?> not found.
</div>
```

```
<?
}

}

?>

<form class="form-inline" method="POST">
<div lass="form-group">
<input type="text" name="gid" placeholder="Enter GID" class="form-
control"></input>
</div>
&nbsp;&nbsp;&nbsp;
<button type="submit" name = "submit1" class="btn btn-info">Road Works</button>
</form>
<hr>
<?php
include "connect.inc.php";
if(isset($_POST['submit1'])){
$gid = $_POST['gid'];

//$sql="SELECT * FROM ways WHERE GID='$gid'";
$sql="update ways set dynamic_cost=length WHERE GID='$gid'";
$result = pg_Exec($db,$sql);

}

?>

<a href="view_all_blocked_route.php" target="_blank">View Blocked Routes</a>
</div>
</div>
<script>

var map = L.map('mapid').setView([27.46735, 89.64445], 13);

var osmUrl = 'http://{s}c.tile.openstreetmap.org/{z}/{x}/{y}.png';
var osmAttrib = 'Map data © OpenStreetMap contributors';
var osm = new L.TileLayer(osmUrl, { attribution: osmAttrib });


var Stamen_Watercolor = L.tileLayer('https://stamen-tiles-
{s}.a.ssl.fastly.net/watercolor/{z}/{x}/{y}.(Language(HTML))', {
attribution: 'Map tiles by <a href="http://stamen.com">Stamen Design</a>, <a
href="http://creativecommons.org/licenses/by/3.0">CC BY 3.0</a> &mdash; Map
data &copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',
subdomains: 'abcd',
minZoom: 1,
maxZoom: 16,
ext: 'jpg'
});

var Esri_WorldImagery =
L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', {
attribution: 'Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community'
});

var baseLayers = {
"Open Street Map": osm,
"Water Color": Stamen_Watercolor,
"Esri World": Esri_WorldImagery

};

var road = L.tileLayer.wms('http://localhost:8088/geoserver/final/wms?', {
layers: 'final:ways_final',
format: 'image/png',
transparent: true,
opacity: 0.7,
});
var tourist_area = L.tileLayer.wms('http://localhost:8088/geoserver/final/wms?', {
layers: 'final:tourist_area_final',
format: 'image/png',
transparent: true,
opacity: 0.7,
});
var health = L.tileLayer.wms('http://localhost:8088/geoserver/final/wms?', {
layers: '	final:health_facilities_final',
format: 'image/png',
transparent: true,
opacity: 0.7,
});
var road_GID = L.tileLayer.wms('http://localhost:8088/geoserver/final/wms?', {
layers: '	final:ways_gid',
format: 'image/png',
transparent: true,
opacity: 0.7,
});

var overlayLayers = {
"Thimphu Road" : road,
"Tourist Area" : tourist_area,
"Health Facilities" : health,
"Thimphu Road GID" : road_GID
};
map.addLayer(osm);
L.control.layers(baseLayers, overlayLayers).addTo(map);


</script>
</body>
</html>
```

Appendix 7

```
<?php
session_start();

if(!isset($_SESSION['username'])){
header("location:../index.php");
exit();
}
?>

<!DOCTYPE html>
<html>
<head>
<title>Leaflet Map</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="shortcut icon" type="image/x-icon" href="docs/images/favicon.ico" />
<link rel="stylesheet"
href="http://localhost/example_routing_on_web/work/bootstrap/css/bootstrap.min.css
">
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.5.1/dist/leaflet.css"
integrity="sha512-
xwE/Az9zrjBIphAcBb3F6JVqxf46+CDLwfLMHloNu6KEQCAWi6HcDUbeOfBIpt
F7tcCzusKFjFw2yuvEpDL9wQ==" crossorigin=""/>
<script src="https://unpkg.com/leaflet@1.5.1/dist/leaflet.js" integrity="sha512-
GffPMF3RvMeYyc1LWMHtK8EbPv0iNZ8/oTtHPx9/cc2ILxQ+u905qIwdpULaqDk
yBKgOaB57QTMg7ztg8Jm2Og==" crossorigin=""></script>
<script
src="http://localhost/example_routing_on_web/work/bootstrap/js/bootstrap.min.js"><
/script>
</head>
<body>
<div class="container-fluid">
<div class="row">
<h6 style="margin:10px;">Logged in as: <?php echo $_SESSION['username']; ?> <a
href="http://localhost/example_routing_on_web/admin/logout.php">Logout</a></h6
>
</div>
<hr>

<table class="table table-hover">
<h3>View all blocked routes</h3>
<tr>
<th>GID</th>
<th>Name</th>
</tr>
```

```
<?php
include "connect.inc.php";

$sql = "SELECT * FROM ways WHERE dynamic_cost=10000000";
$result = pg_Exec($db, $sql);
$num=pg_num_rows($result);
$i = 0;


while($i < $num){
echo "<TR>";
echo "<TD>";
echo pg_Result($result, $i, "gid");
echo "</TD>";

echo "<TD>";
echo pg_Result($result, $i, "name");
echo "</TD>";
echo "</TR>";
$i++;
}
?>
</table>
</div>


</body>
</html>
```

REFERENCES

Agrawal, S., & Gupta, R. D. (2014). *Development and Comparison of Open Source based Web GIS Frameworks on WAMP and Apache Tomcat Web Servers* (Vol. XL-4). https://doi.org/10.5194/isprsarchives-XL-4-1-2014

Bendib, A., Hadda, D., & Kalla, M. (2016, 05/01). Application of Webgis in the development of interactive interface for urban management in Batna City. *Journal of Engineering and Technology Research, 8*, 13-20. https://doi.org/10.5897/JETR2015.0579

Bhutan, T. C. o. (2018). *BHUTAN TOURISM MONITOR 2018*. P. o. t. T. C. o. Bhutan.

Chandniha, S., Goel, M. K., Rathore, D. S., Arora, M., & Gupta, R. (2017). The application of openlayers on web gis for spatio-temporal data publishing. In (pp. 48-72).

Choosumrong Sittichai, H. C., Raghavan Venkatesh and Fenoy Gerald. (2019). *Development of optimal routing service for emergency scenarios using pgRouting and FOSS4G*. https://doi.org/10.1007/s41324-019-00248-2

Choosumrong, S., Raghavan, V., & Bozon, N. (2012, 09/01). Multi-Criteria Emergency Route Planning Based on Analytical Hierarchy Process and pgRouting. *Japan society of Geoinformatics, 23*, 159-168. https://doi.org/10.6010/geoinformatics.23.159

Choosumrong Sittichai, R., Venkatesh,Realini and Eugenio. (2010). *Implementation of dynamic cost based routing for navigation under real road conditions using FOSS4G and OpenStreetMap*.

Chunithipaisan, S., Supavetch and Soravis. (2010). *The Development of Web Processing Service Using the Power of Spatial Database*. https://doi.org/10.1109/ICETET.2009.222

Deepa, G., Kumar, Priyank ,Angamuthu,Manimaran,Rajakumar k, and Krishnamoorthy. (2018). *Dijkstra Algorithm Application: Shortest Distance between Buildings*

(Vol. 7). https://doi.org/10.14419/ijet.v7i4.10.26638

Feng, T., Bi, J., & Hu, H. (2011). *OpenRouter: OpenFlow extension and implementation based on a commercial router*. https://doi.org/10.1109/ICNP.2011.6089045

Fischer, M. (2004). GIS and network analysis. In (pp. 391-408).

Haitao, J., Jin, F., Qing, H., He, Z., & Xue, Y. (2019, 07/31). Measuring Public Transit Accessibility Based On Google Direction API. *The Open Transportation Journal*, *13*, 93-108. https://doi.org/10.2174/1874447801913010093

Han, Y. (2018). *Digital mapping for cartographic narrative in Dublin using Leaflet Digital mapping for cartographic narrative in Dublin using Leaflet for R*.

Hendawi, A., Rustum, A., Ahmadain, A., Hazel, D., Teredesai, A., Oliver, D., Ali, M., & Stankovic, J. (2017). *Smart Personalized Routing for Smart Cities*. https://doi.org/10.1109/ICDE.2017.172

Kumari Pritee, G. R. D. (2017). Identification of Optimum Shortest Path using Multipath Dijkstra's Algorithm Approach. *International Journal of Advanced Remote Sensing and GIS.*

Lange, N., & Plass, C. (2008, 01/01). WebGIS with Google Maps.

Language(HTML), H. M. Retrieved June from https://www.w3schools.com/html

Leaflet. Retrieved June from http://leafletjs.com

Lizardo, L., & Davis Jr, C. (2017). *A PostGIS extension to support advanced spatial data types and integrity constraints*. https://doi.org/10.1145/3139958.3140020

Longo, M., & Roscia, M. (2014). *Sustainable transportation application for smart mobility*. https://doi.org/10.1109/SPEEDAM.2014.6872112

Mooney, P., & Minghini, M. (2017). A review of OpenStreetMap data. In (pp. 37-59). https://doi.org/10.5334/bbf.c

OpenLayers. Retrieved July from http://openlayers.org

OpenStreetMap. Retrieved January from http://www.openstreetmap.org/

PHP. Retrieved June from https://www.php.net

Pritee, K., & Garg, R. (2017). *Identification of Optimum Shortest Path using Multipath Dijkstra's Algorithm Approach* (Vol. 6). https://doi.org/10.23953/cloud.ijarsg.321

Project, p. *pgRouting Project*. Retrieved December from www.pgrouting.org/

Prokofyeva, N., & Boltunova, V. (2017, 12/31). Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. *Procedia Computer Science*, *104*, 51-56. https://doi.org/10.1016/j.procs.2017.01.059

QGIS. Retrieved January from http:qgis.org/en/site/

Road Safty and Transport Authority, M. o. I. a. C., Bhutan. (June 2019). *Annual Info-Comm and Transport Statistics 2019*. www.moic.gov.bt

S. Singh, P., B. Lyngdoh, R., Chutia, D., Saikhom, V., Kashyap, B., & Sudhakar, S. (2015). *Dynamic shortest route finder using pgRouting for emergency management* (Vol. 7). https://doi.org/10.1007/s12518-015-0161-4

Shaw, S., & Gurram, D. (2015, 05/01). A Note on Computational Approach to Travelling Sales Man Problem. *International Journal of Computer Applications (0975 – 8887) Volume 115 – No. 8, April 2015*, *115*, 975-8887. https://doi.org/10.5120/20174-2374

Singh, P., Lyngdoh, R., Chutia, D., Saikhom, V., Kashyap, B., & Sudhakar, S. (2015, 07/23). Dynamic shortest route finder using pgRouting for emergency management. *Applied Geomatics*, *7*. https://doi.org/10.1007/s12518-015-0161-4

Xie Dexiang, Z. H., Yan Lin,Yuan Si and Zhang Junqiao. (2012). *An improved Dijkstra algorithm in GIS application*.

Zhang, L., & He, X. (2012). Route Search Base on pgRouting. In (Vol. 115, pp. 1003-1007). https://doi.org/10.1007/978-3-642-25349-2_133