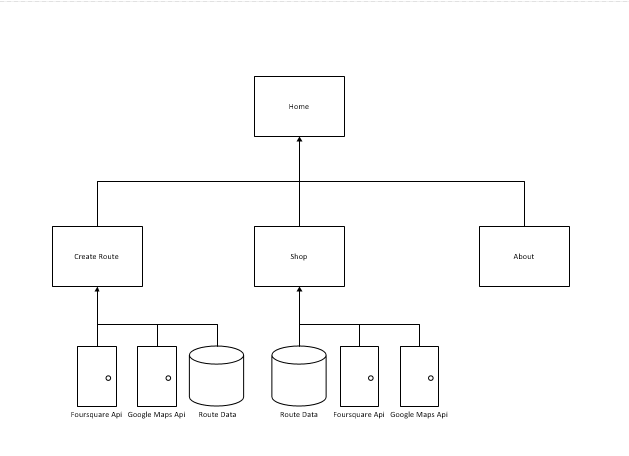
**Midterm - Hackathon**

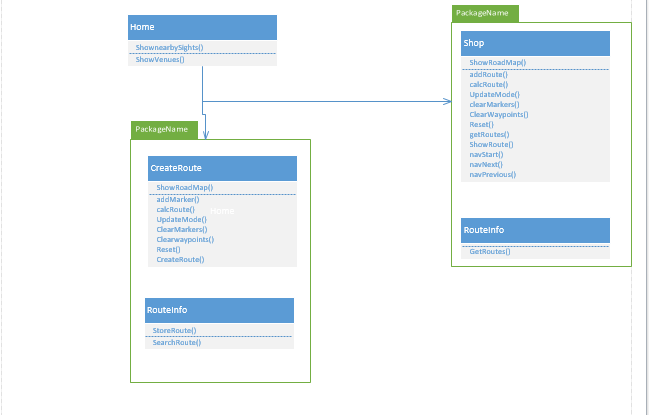
**Vinutha Nuchimaniyanda**

**Student Id: 16147487**

1. **Design**
2. System Architecture Diagram



1. Class Diagram

****

1. **Features Implemented**
2. **SOAP WebService**

Tour Buddy Services

1. StoreRoute – service method to store route and route waypoint information.
2. GetRoutes – service method to retrieve all routes stored in the application.
3. SearchRoutes – service method to search routes by route name.

**Existing API’s**

1. Geolocation & Geocode for google maps
2. Foursquare API for venues
3. Yahoo API for layout
4. **Databases**

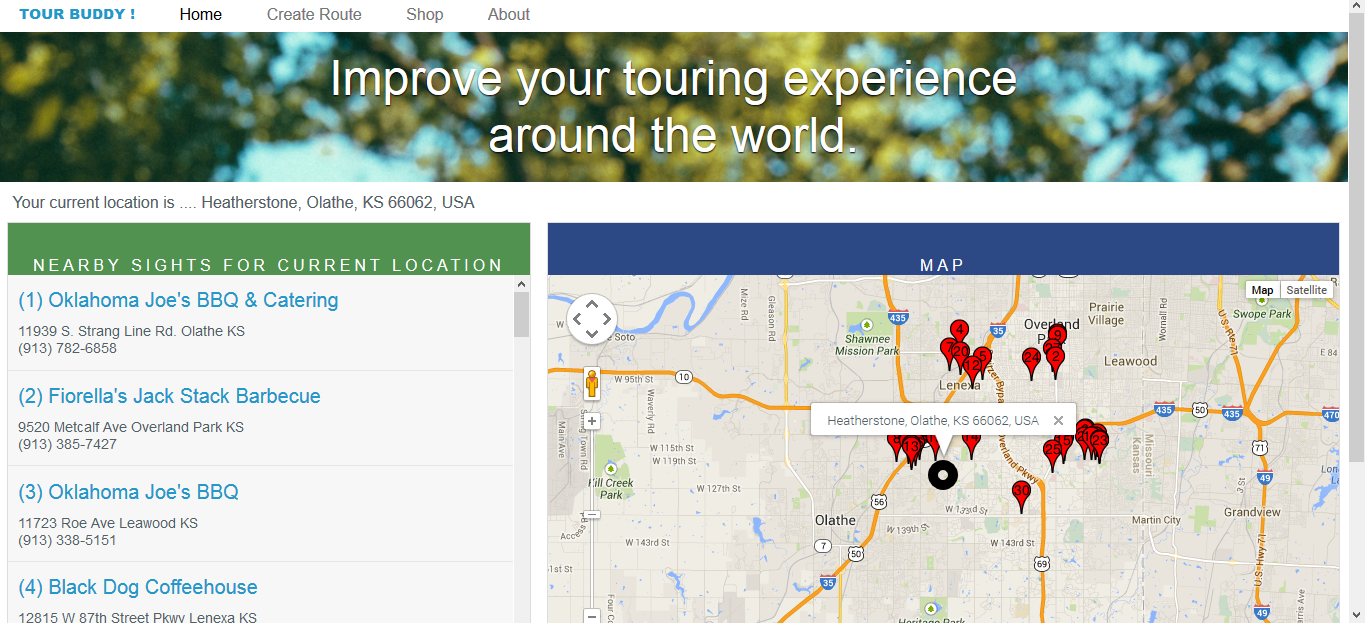
Tour Buddy Databases

1. Route table with all the route information ( mode, imageURL, description)
2. RouteWaypoints with latitude and longitude info
3. **Screenshots and Description of Features**

**Tour Buddy**

Tour buddy is a web application that uses Foursquare venues API to display the places around user’s current location. Current location of the user is calculated using geocode of Google maps Api. This Application can be used to create geographic routes around the world from the places registered at Foursquare. Users can create their own routes. Directions are available for these routes. Few routes are available under shop that can be purchased by the user. While creating routes, user has the option to add images and description about the various checkpoints that can later be shared with others.

Below is the home page of Tour Buddy that display’s the current location on the map with the list of nearby businesses and sights. The layout of the screen is burrowed from Yahoo API that uses pure css for styling. Google maps API with geolocation to display current location of the user. Markers on the map are numbered displaying all the available spots in Olathe, Kansas. The Nearby sights that are displayed on the left hand side uses FourSquare Venue API.

****

Major functionalities of this application are:

1. Create Route- allows user to create the route on the map that is saved in the database that can later be retrieved.
2. Shop – Lists all the available routes and user also has option to navigate through the routes.

**Create Route**

Create route contains

Mode of transportation which can Bicycle, Walking and Driving

Route Name for name of the route

Image for the url of the image to be uploaded

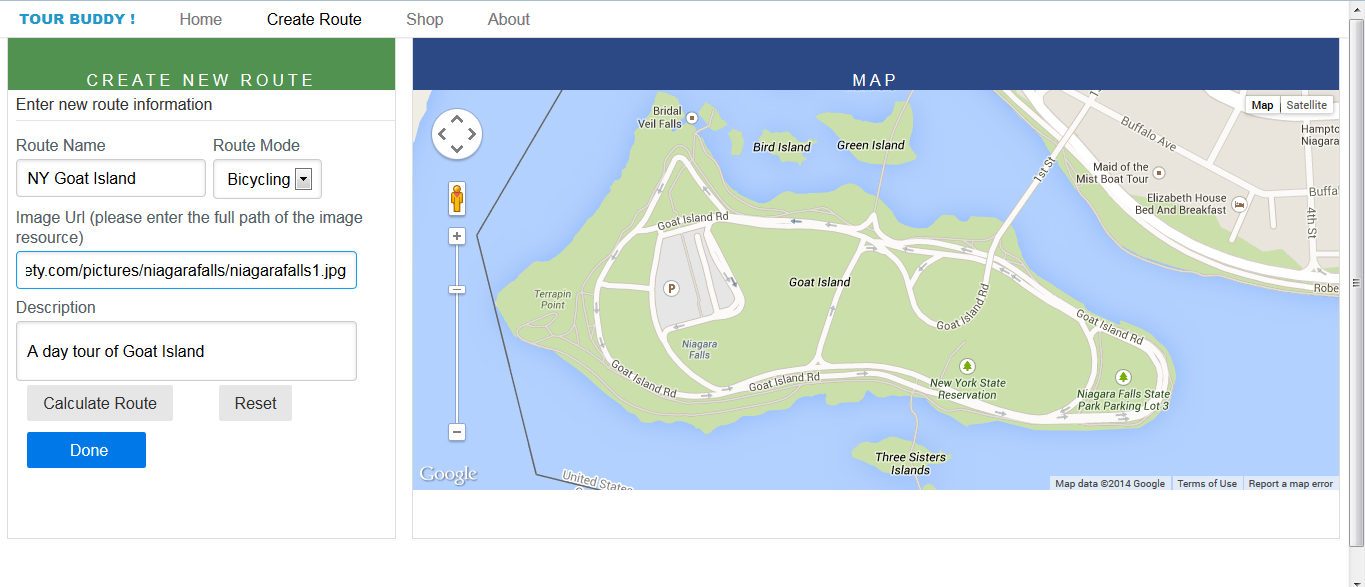
Description for notes about the route

User can click on the map to create markers at various places for checkpoints and also add description, Image URL and Title for the checkpoint. Once he is done with adding markers at various location, he can click calculate route which will display route on the map and also directions from one checkpoint to another. User can choose any of the three available mode of transportation which will calculated using Google direction renderer API. After entering all the information, when user clicks Done. The filled information is saved in the database.

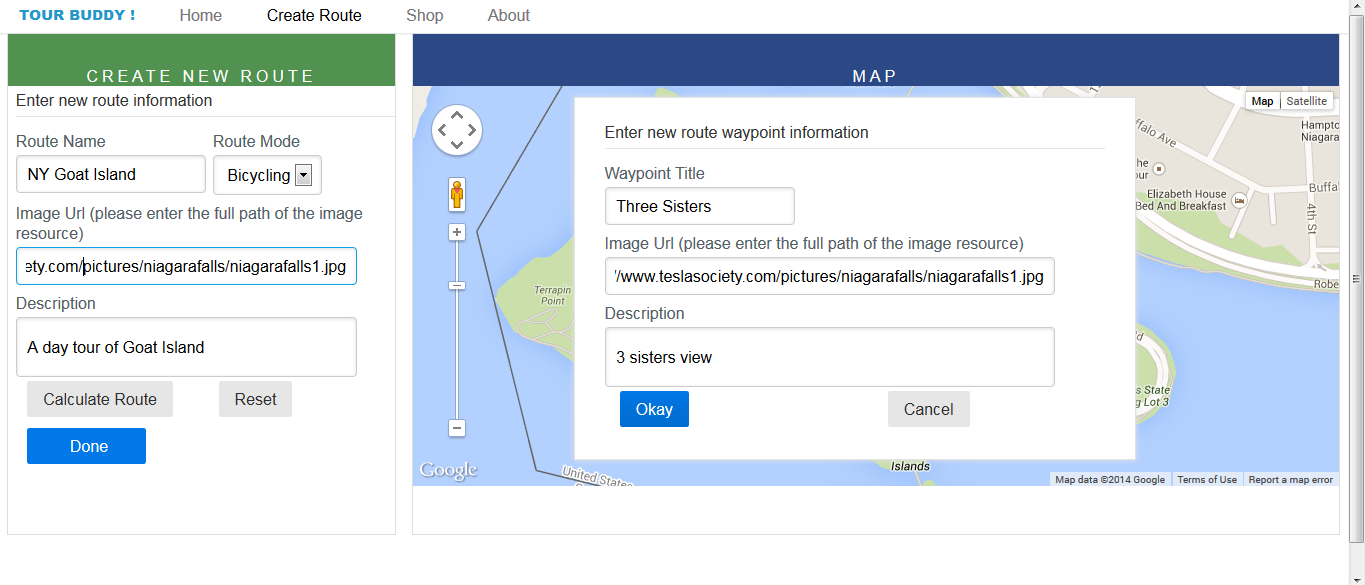
There is Reset button to Start over the process.

Below are the screenshots

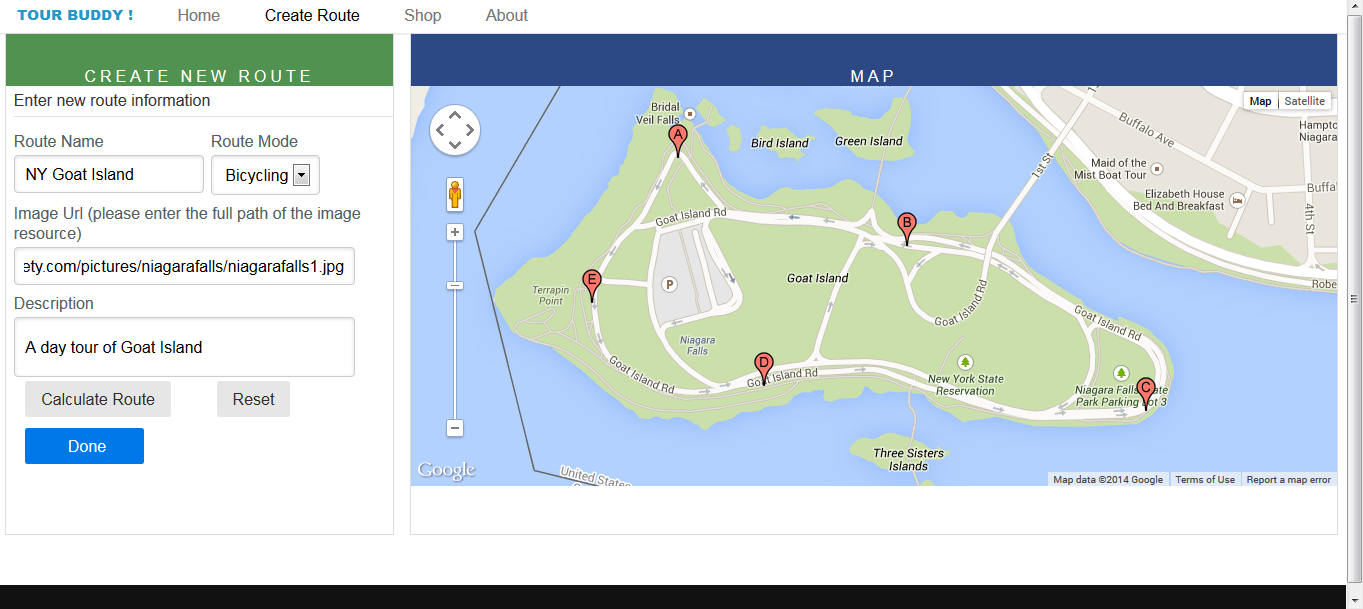
Create route



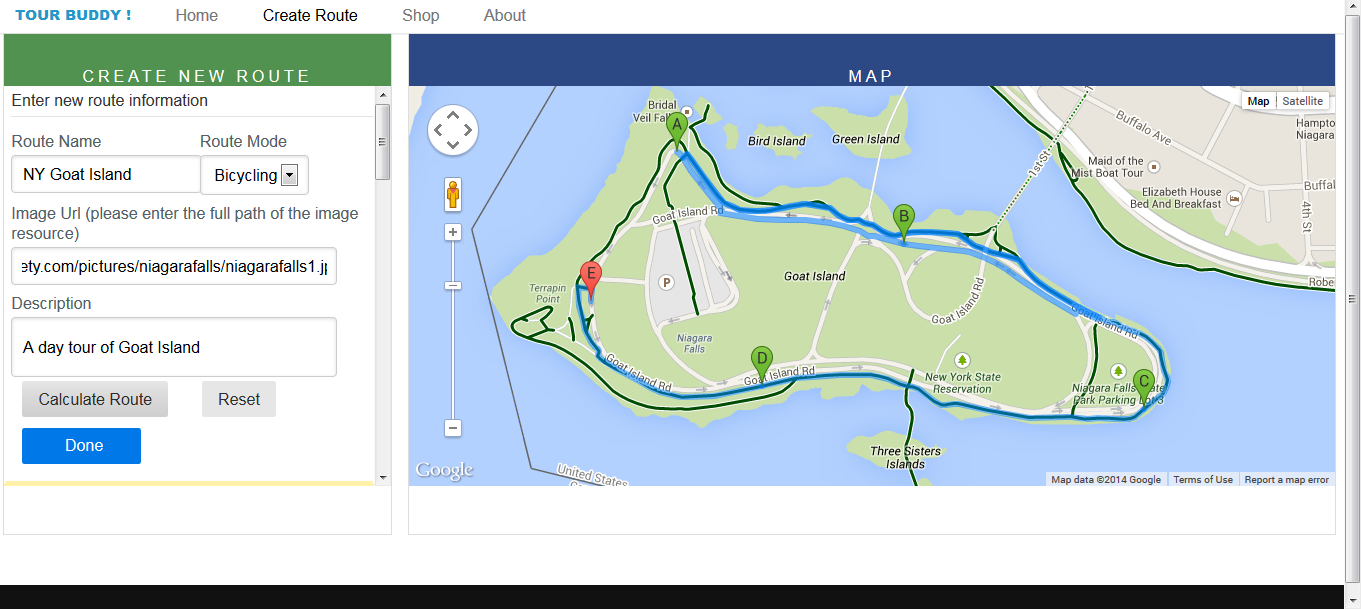
When clicked on the marker shows the option to add description and image at the marker.



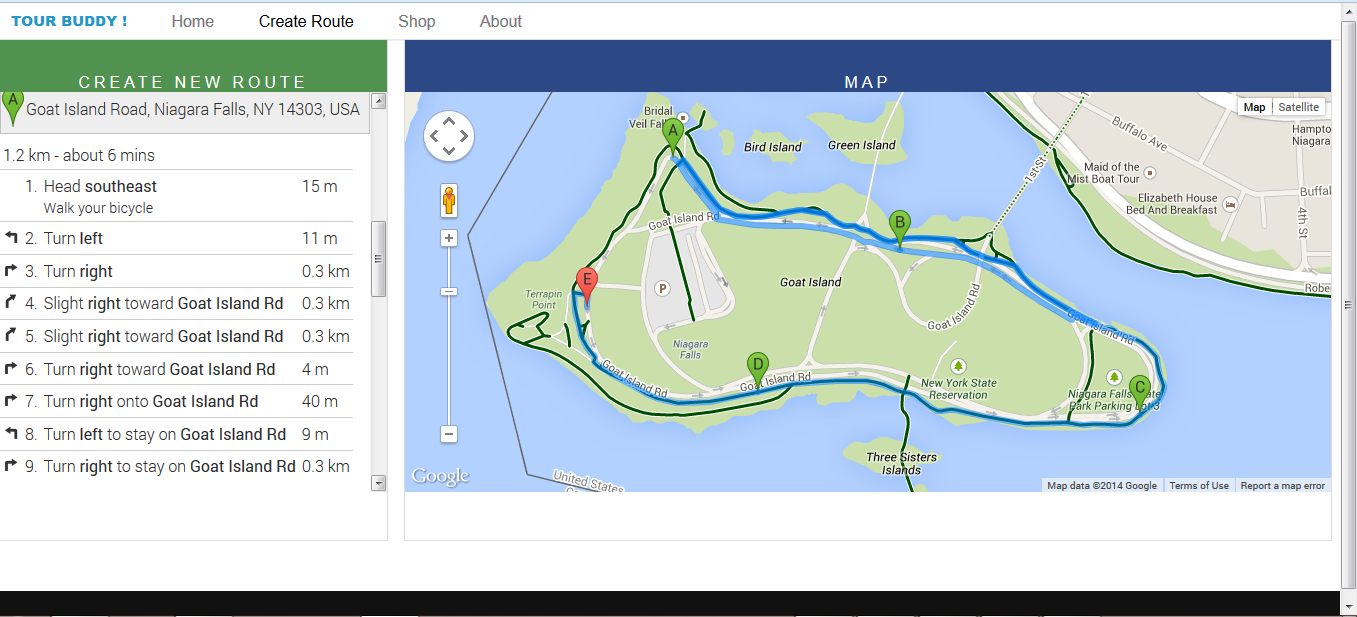
Markers at various locations.



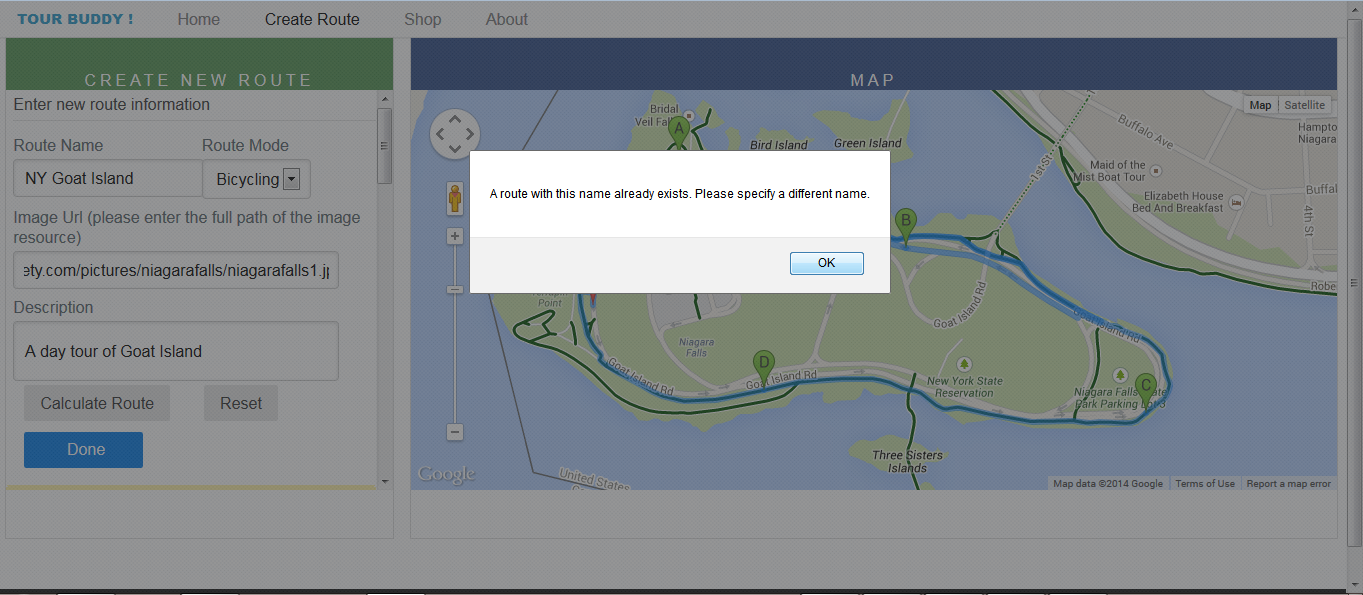
Calculate route after adding markers



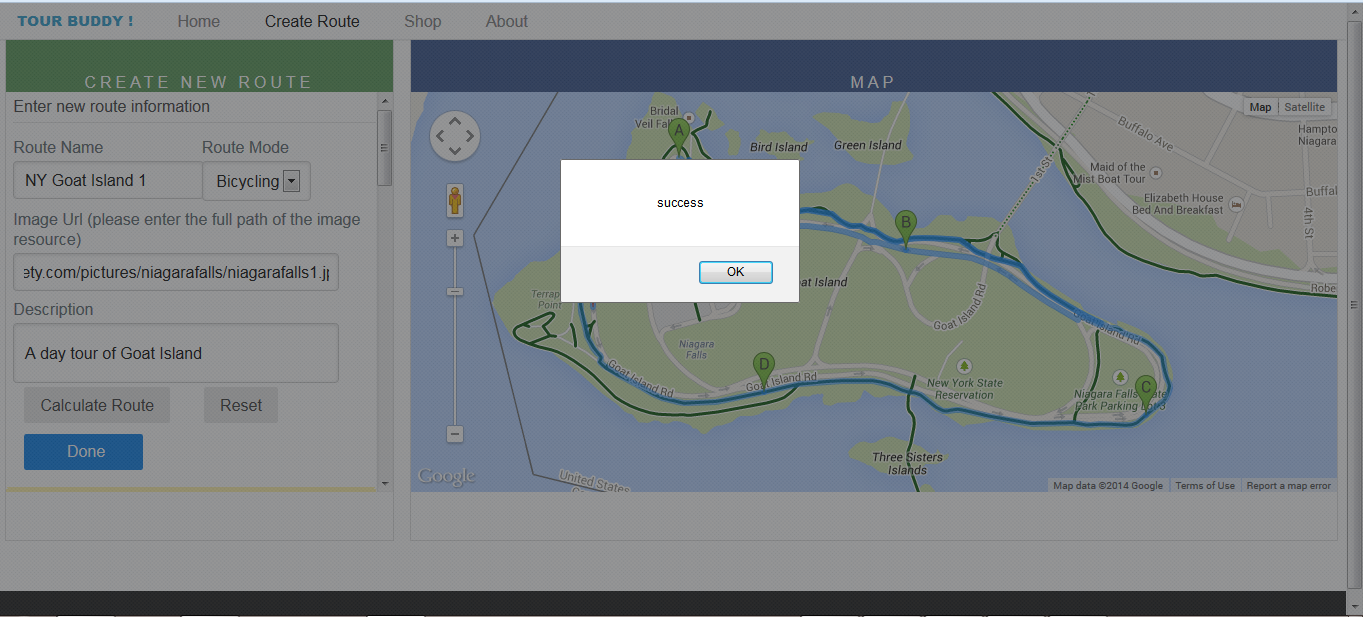
Directions displayed after calculating the route.



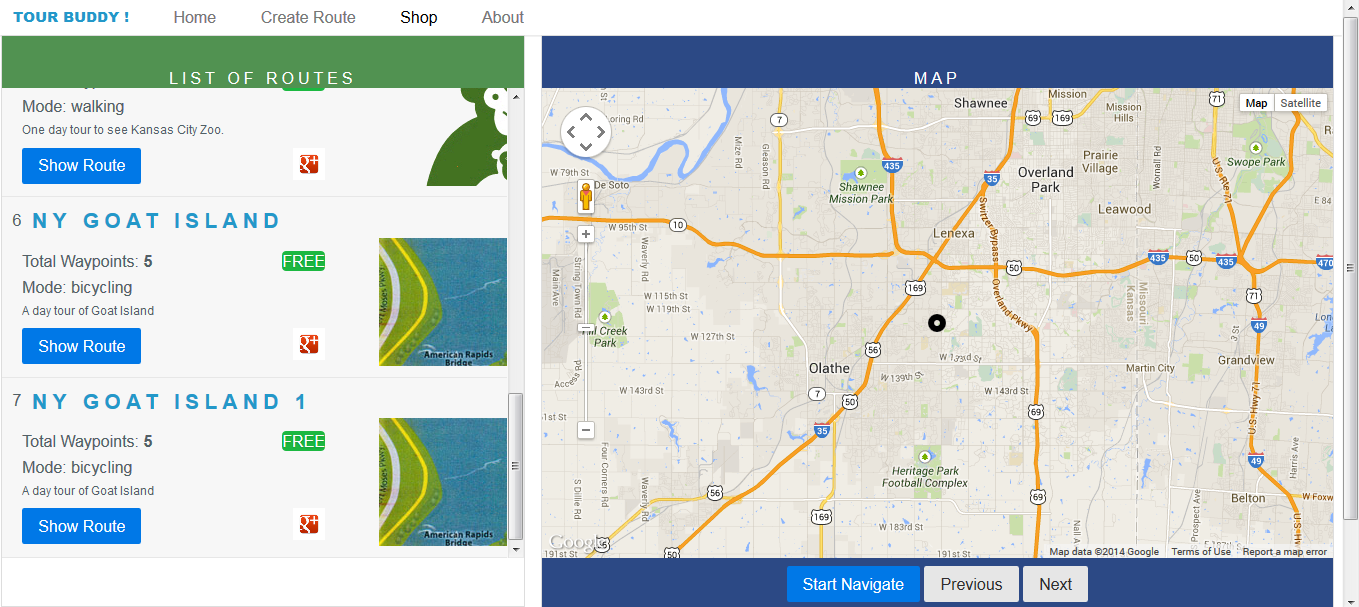
User message for the same route name.



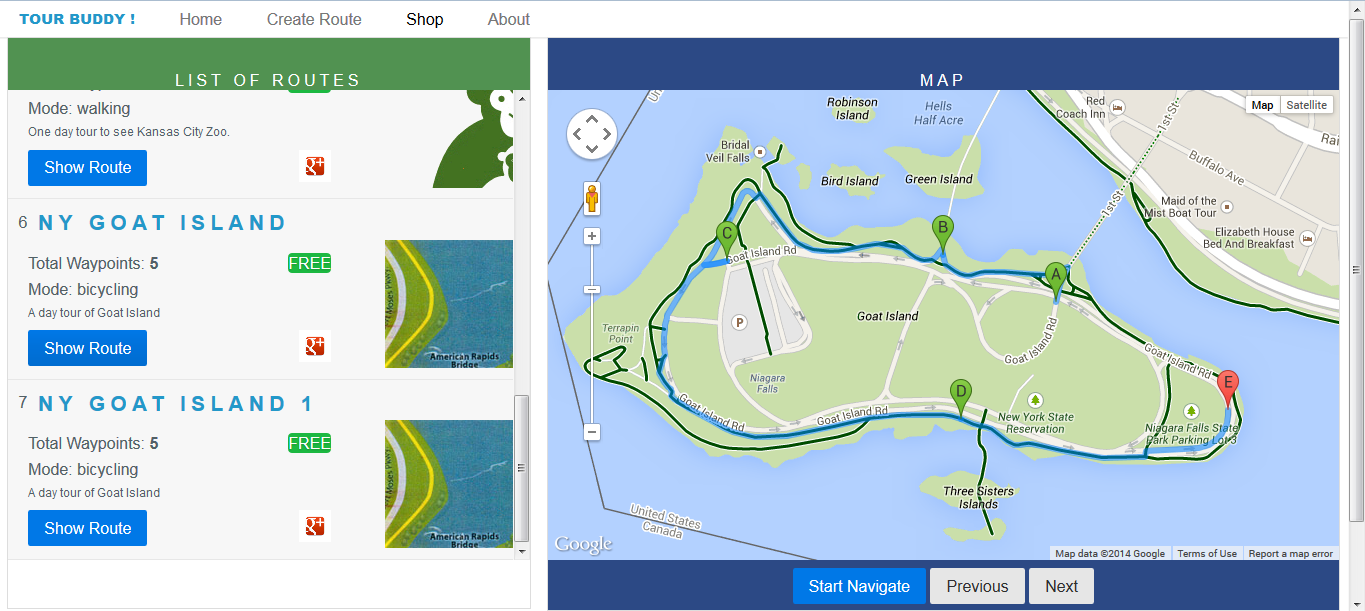
Success message after the record is added to the database



Shop has list of routes available for the user to purchase. Few routes are free and user has to buy the ones that are not free. User can navigate through different checkpoints here by clicking next and previous buttons

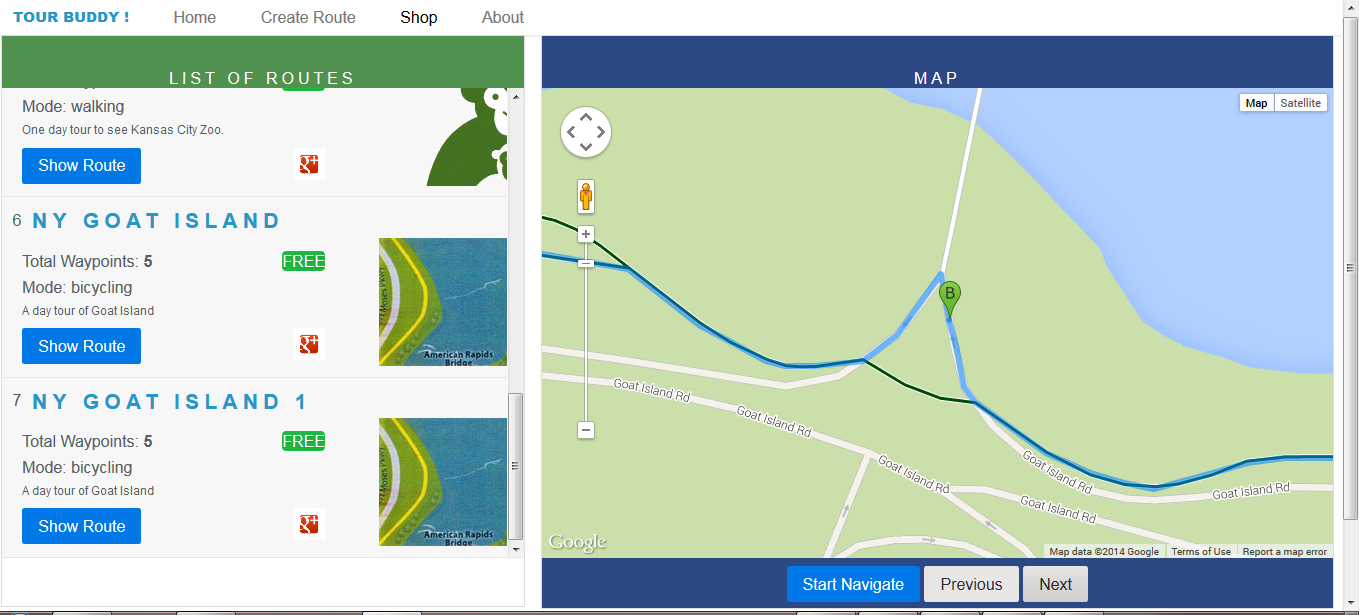


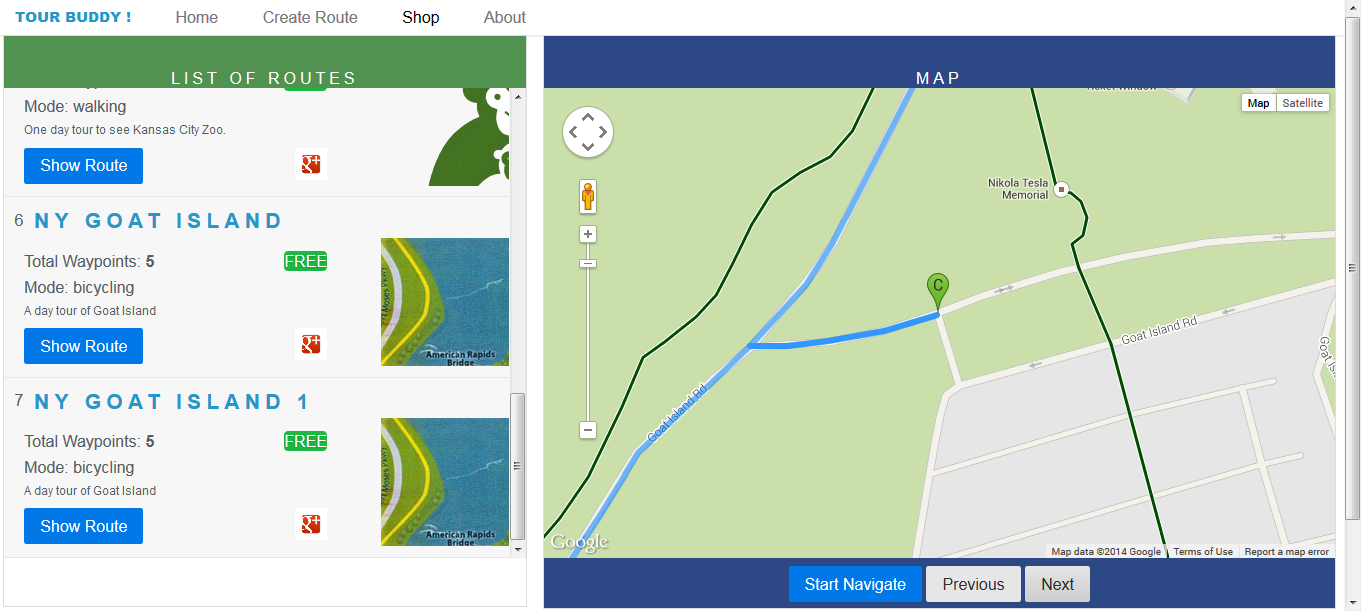
The Route that was created earlier is displayed on the map when user clicks Show Route button

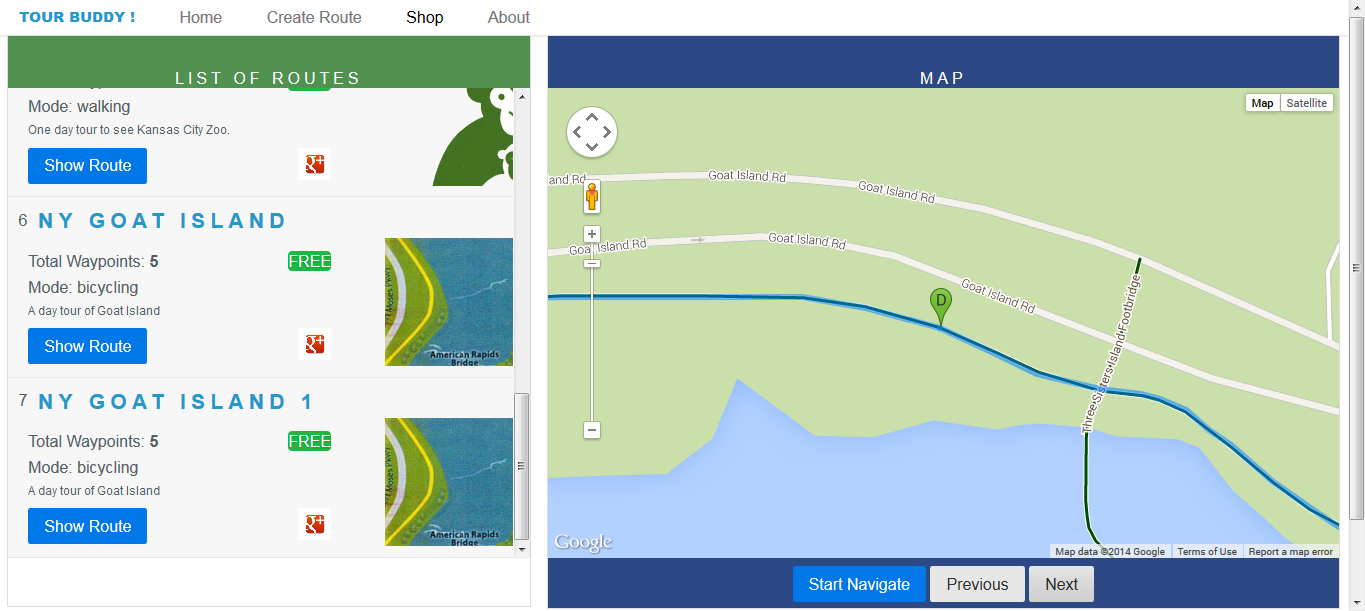


Navigating through checkpoints with next button.

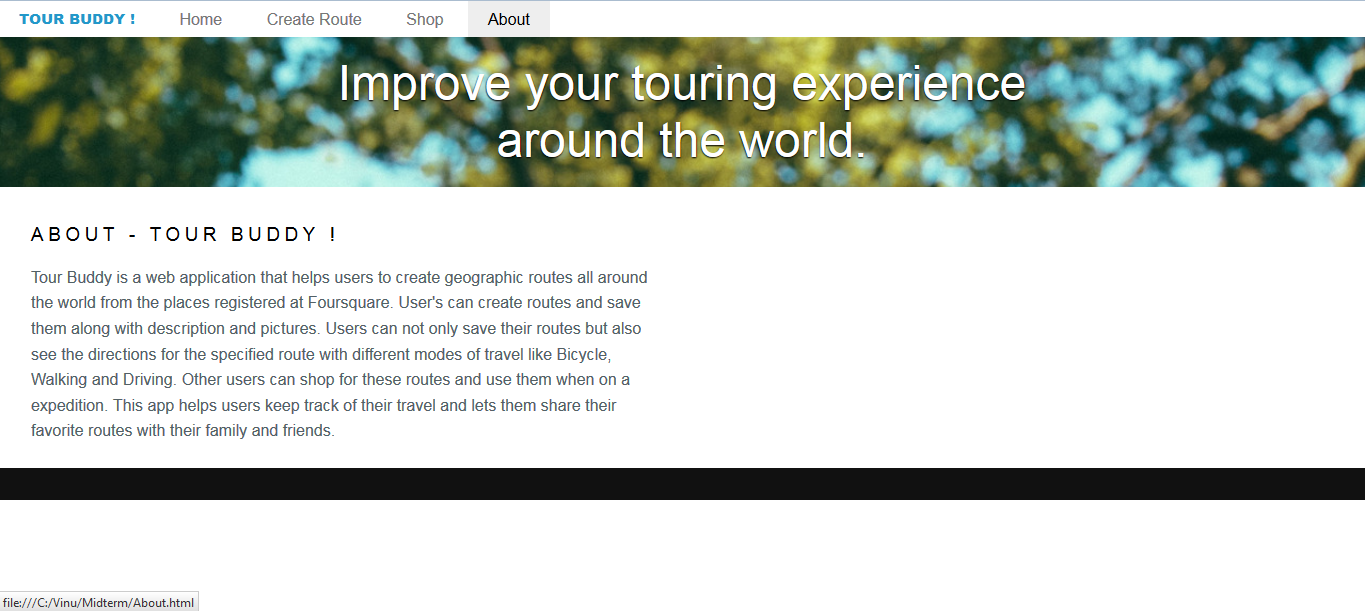




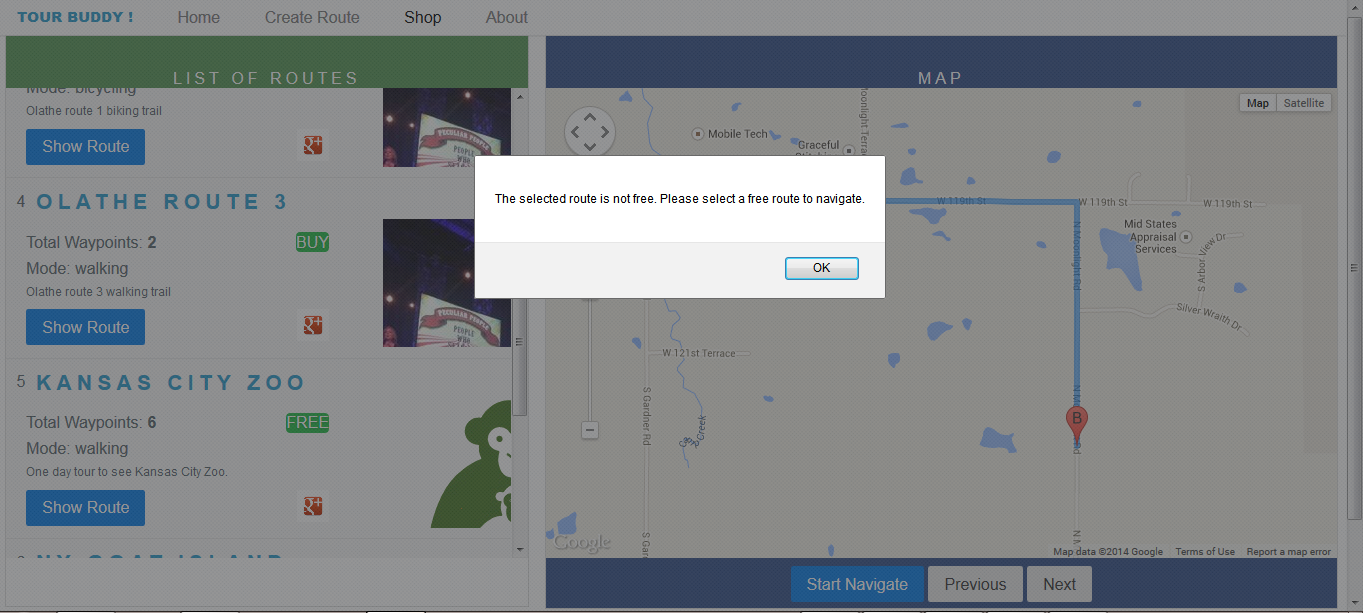




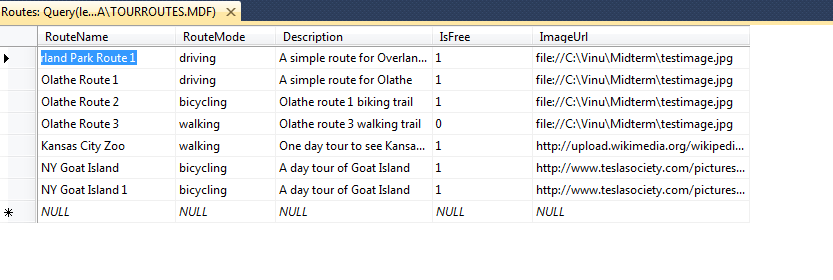
About page that displays information about the project



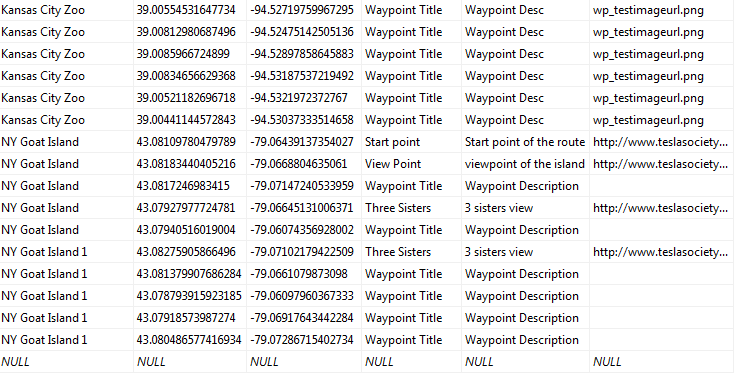
Message displayed when user open the BUY route.



Below are the screenshots for the routes added to the database.



Marker information added to the database.



1. **Web Service and Web Site URL**
2. <http://localhost:50183/WebService1.asmx?op=SearchRoutes>

### SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

POST /WebService1.asmx HTTP/1.1

Host: localhost

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://tempuri.org/SearchRoutes"

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<SearchRoutes xmlns="http://tempuri.org/">

<routename>string</routename>

</SearchRoutes>

</soap:Body>

</soap:Envelope>

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<SearchRoutesResponse xmlns="http://tempuri.org/">

<SearchRoutesResult>

<RouteInfo>

<RouteName>string</RouteName>

<RouteMode>string</RouteMode>

<Description>string</Description>

<IsFree>boolean</IsFree>

<ImageUrl>string</ImageUrl>

<MarkerList>

<Position xsi:nil="true" />

<Position xsi:nil="true" />

</MarkerList>

</RouteInfo>

<RouteInfo>

<RouteName>string</RouteName>

<RouteMode>string</RouteMode>

<Description>string</Description>

<IsFree>boolean</IsFree>

<ImageUrl>string</ImageUrl>

<MarkerList>

<Position xsi:nil="true" />

<Position xsi:nil="true" />

</MarkerList>

</RouteInfo>

</SearchRoutesResult>

</SearchRoutesResponse>

</soap:Body>

</soap:Envelope>

1. <http://localhost:50183/WebService1.asmx?op=StoreRoute>

### SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

POST /WebService1.asmx HTTP/1.1

Host: localhost

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://tempuri.org/StoreRoute"

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<StoreRoute xmlns="http://tempuri.org/">

<data>string</data>

</StoreRoute>

</soap:Body>

</soap:Envelope>

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<StoreRouteResponse xmlns="http://tempuri.org/">

<StoreRouteResult>string</StoreRouteResult>

</StoreRouteResponse>

</soap:Body>

</soap:Envelope>

1. <http://localhost:50183/WebService1.asmx?op=GetRoutes>

**SOAP 1.1**

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

POST /WebService1.asmx HTTP/1.1

Host: localhost

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://tempuri.org/GetRoutes"

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<GetRoutes xmlns="http://tempuri.org/" />

</soap:Body>

</soap:Envelope>

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<GetRoutesResponse xmlns="http://tempuri.org/">

<GetRoutesResult>

<RouteInfo>

<RouteName>string</RouteName>

<RouteMode>string</RouteMode>

<Description>string</Description>

<IsFree>boolean</IsFree>

<ImageUrl>string</ImageUrl>

<MarkerList>

<Position xsi:nil="true" />

<Position xsi:nil="true" />

</MarkerList>

</RouteInfo>

<RouteInfo>

<RouteName>string</RouteName>

<RouteMode>string</RouteMode>

<Description>string</Description>

<IsFree>boolean</IsFree>

<ImageUrl>string</ImageUrl>

<MarkerList>

<Position xsi:nil="true" />

<Position xsi:nil="true" />

</MarkerList>

</RouteInfo>

</GetRoutesResult>

</GetRoutesResponse>

</soap:Body>

</soap:Envelope>

1. **Github URL**

[**https://github.com/vn539/Midterm**](https://github.com/vn539/Midterm)

1. **Limitations**
2. Foursquare Api Venue includes all the places around the Geocode, could not be filtered only for Tourist spots.
3. Finding the Best Route among the available routes
4. No sharing functionality available through social networks
5. **References**
6. <https://developer.foursquare.com/overview/tutorial>
7. <http://yui.yahooapis.com/pure/0.4.2/pure.css>
8. <https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false>
9. <https://code.jquery.com/jquery-1.10.2.js>
10. <https://api.foursquare.com/v2/venues/explore?client_id=4SL20OZILKSDVOTWOGYB2MRH4KREAIB5RK5IIEHLDKWZ0OZO&client_secret=ZULYPUVFKIQOHZ0PRS1JL0JZ4ZBZ10H4ST5B1HAMIWYCGDGD&v=20130815&ll>