Proceedings of the National Conference
On Undergraduate Research (NCUR) 2020
Montana State University, Bozeman, MT
March 26-28, 2020

A Safer Route: Integrating and Visualizing Collision Data for Trip Planning

Kokoutse Xavier Doh
School of Computer Science and Technology
Kean University
Union, NJ 07083 USA

Faculty Advisor: Daehan Kwak

Abstract

Technology significantly evolves over time in a way that almost everyone relies on GPS-based navigation systems in their daily commute to navigate from one place to another. To select the best possible way for a user to get from location A to B, some decision factors include the ETA (Estimated Time Arrival) and/or the shortest distance needed to get to the destination. However, in the route selection decision-making process, other factors such as the distribution of "accidents that occurred along a route" are not incorporated. For instance, studies show that women and elders have more chances to select safer routes than to faster routes. This study integrates navigation systems to visually display a real time risk level of accidents that occurred, including the type of accident frequently involved on the selected road to keep the drivers notified and attentively pay attention to the road laws in order to safely get to their destinations, or even avoid those routes. In detail a web-based navigation system is developed using Google Maps Platform along with data on Motor Vehicle Collision Reports for New York capable of displaying this information. The system has two main functions: a data visualization of previous collisions based on their location and a display of collision along the selected road of the map. On top of that, the system counts each collision to make a report notifying about their frequency on each street. This information system is developed based on the advanced web development tools including JavaScript and PHP.

Keywords: Estimated Time Arrival (ETA), Intelligent Transportation System (ITS), Traffic Collision Analysis.

1. Introduction

In general, when selecting or planning a route, our decision making depends on factors such as ETA, the Estimated Time Arrival, the size of traffic, the distance, and the safety of the route. In the research regarding traffic behavior [1], men are more risk-takers, meaning they prefer shorter time versus safer routes whereas, women and elders are risk-averse, meaning they prefer safer routes versus shorter time. However, in today's navigation system, there is no User Interface that can show the safeness (or risk level) of routes that are of interest. Several previous data showed that being aware of your surroundings can drastically reduce safety incidents, therefore by providing such system, to visually display a risk level on routes of interest based on accident history data, this can reduce traffic incidents.

2. Related Work

Providing various information to the user regarding route selection provides the user with an option to select routes based on their preference. Previous research interest in traffic and route trip analysis have showed great understandings in how route guidance systems operate. Upon comparison of several scenarios to get users to their destinations, the navigator system selects the best ETA route trip. As limitation occurs in the route choice which potentially brings

uncertainty to the drivers' experience, the use of visual traffic information for pre-trip route choice were proposed [2, 3]. The use of post-trip information, the time it took to take a route on non-chosen routes was proposed as well [4].

3. Data Mining and Analysis

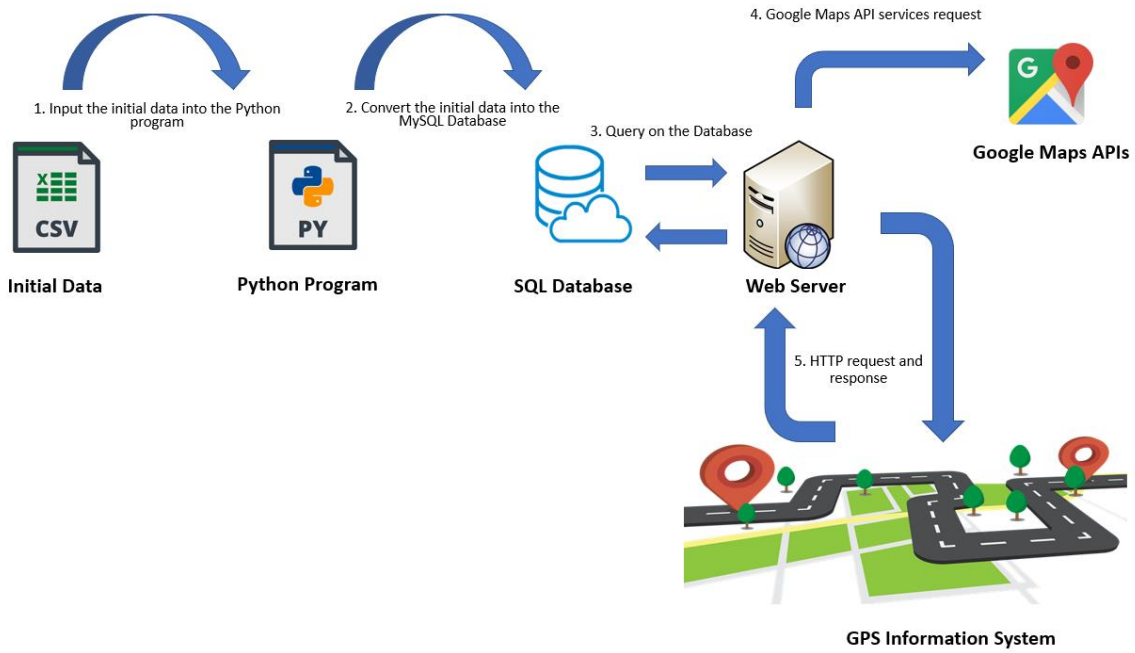

Figure 1: Data flaw diagram.

The figure above consists of how the initial data is transformed throughout the process of becoming part of the designed GPS information system. Initially, the available data as seen in figure 2 is used by a Python program consisting of converting all rows into a MySQL table. The Python script can be seen in figure 3. Once executed, it selects the CSV file containing all recorded crashes data, connect to the Localhost, and the MySQL database "nycrashresearch", the execute an SQL query that can be seen below in figure 4, which create the table "crashrecords" including its rows and inserting each line records. Then PHP and MySQL are used to perform queries on the database to load and display collision data on the website. The figure 5 and 7 show this process. Based on the need of implementing a Google map, the data coming from the database should be labeled on a map, therefore, Google maps APIs and JavaScript facilitates this process. The figure 6 and 8 give a brief view of this implementation, displaying not only a map content but also the driver route trip.

4. Materials and Methods

4.1 Data Set

As the need occurred, a data set is required in this project. This data should be composed of recent collision history. The history would then be available for the user to visualize in real time, browse and retrieve crucial information from it. Some of these information are: a date and time for collisions, the exact coordinate such as latitude and longitude, the zip code, the address including the intersection of roads if applicable, the number of victims who died or were injured and also the type of vehicles involved into these collision. The data should also be able to communicate directly with the web development tools including PHP, MySQL, and JavaScript for efficient query operations.

This inquiry led to the discovery of NYC collision data[5, 6]. The data is composed of all the input need required to accomplish this research. Real NYC collision reports were classified in a CSV file format displaying all collisions occurred between 2010 and 2020. On top of that, the reason behind the collisions were also disclosed.

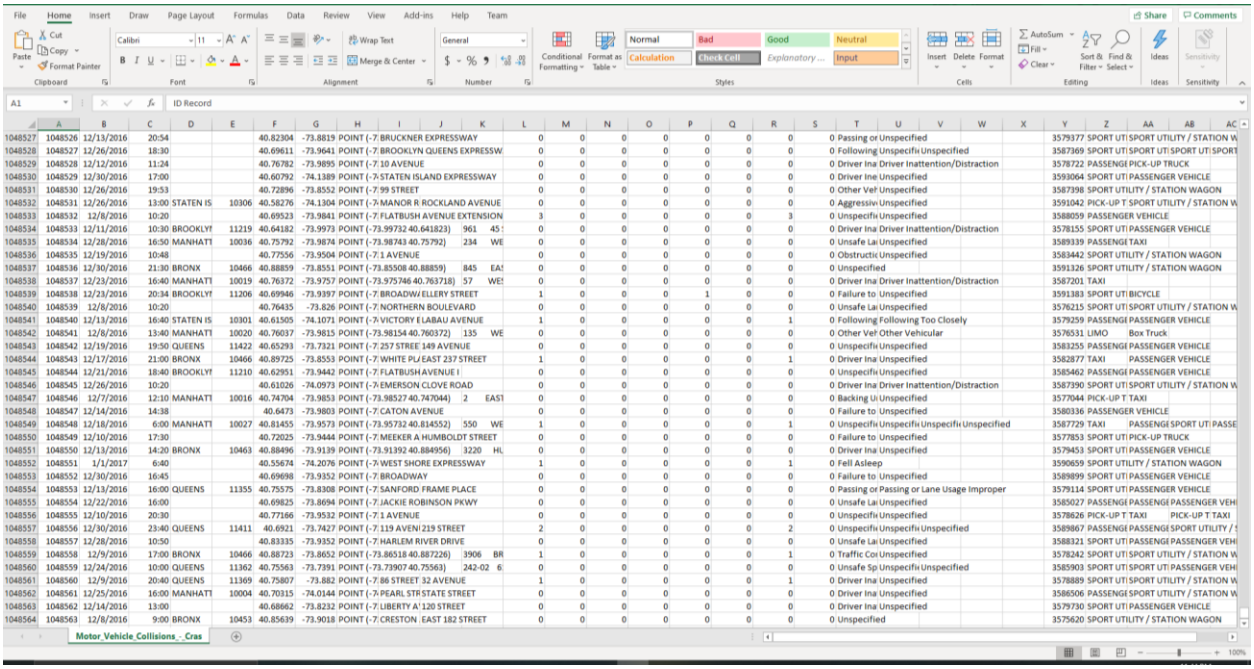

Figure 2: NYC collision data view from Excel.

In addition, Google Maps APIs [4, 7] are involved for specific purposes. The platform should display a map on which users can plan and visualize their trip including the steps, time and distance required to get to destination. In addition, the platform should also display specific areas where collision occurred along the road trip. These are crucial since the frequent collision that happens on each road of the user trip plan will also be visualized, to notify drivers. Therefore, in this research, some Google Maps Platform APIs are used, such as: Maps JavaScript API [8], Direction API [9], Places API [10], and the Geolocation API [11].

4.2 Data manipulation

As the input data is chosen, it should be set to be accessible by the platform. The format the of the data is a CSV file. Therefore, it requires to be converted it into a MySQL [12] database to facilitate the queries. To do so, PyCharm [13], an integrated development environment specialized in Python development, is used.

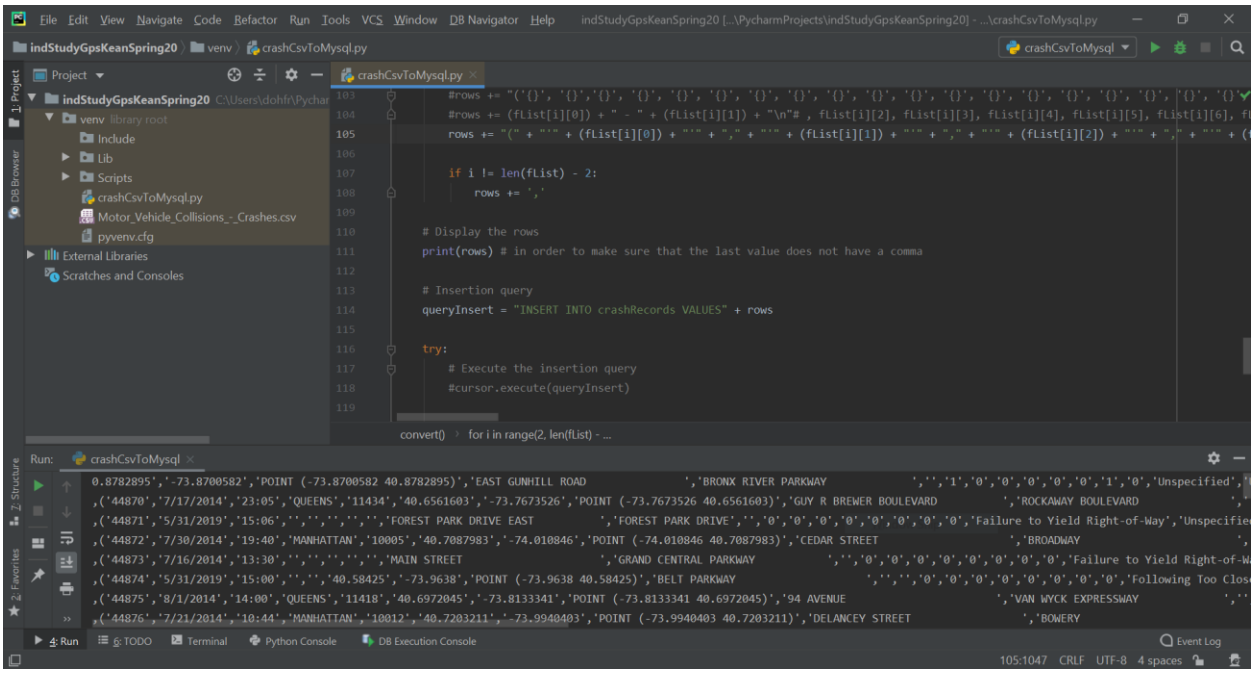

Figure 3: View of the Python script in PyCharm which converts the input data into MySQL database

Using the Apache XAMP server to host the database and importing PYMYSQL in the python source code, the database named “nycrashresearch” is created, which will host the table in which all content of the CSV collision data will be loaded. To do so, the following command was used.

```
import pymysql # Open connection to the DB

db = pymysql.connect("localhost", "root", "", "nycrashresearch")
```

As the CSV file contains about 29 columns, the columns should be created at first then load the two million collision data records inside. The following Python and SQL instructions used to create the columns.

```
queryCrashRecordTable = """CREATE TABLE crashRecords( idCrashRecord int not
null, crashDate varchar(255) not null, crashTime varchar(255) not null, borough
varchar(255) not null, zipCode int, latitude float(10, 6), longitude float(10,
6), location varchar(255) not null, onStreetName varchar(255) not null,
crossStreetName varchar(255) not null, offStrretName varchar(255) not null,
numPersonInjured tinyint, numPersonKilled tinyint, numPredestrianInjured
tinyint, numPredestrianKilled tinyint, numCyclistInjured tinyint,
numCyclistKilled tinyint, numMotoristInjured tinyint, numMotoristKilled
tinyint, contributingFactorVehicle1 varchar(255) not null,
contributingFactorVehicle2 varchar(255) not null, contributingFactorVehicle3
varchar(255) not null, contributingFactorVehicle4 varchar(255) not null,
contributingFactorVehicle5 varchar(255) not null, collisionID int,
vehicleTypeCode1 varchar(255) not null, vehicleTypeCode2 varchar(255) not null,
vehicleTypeCode3 varchar(255) not null, vehicleTypeCode4 varchar(255) not null,
vehicleTypeCode5 varchar(255) not null, primary key(idCrashRecord) )"""
```

Execute the table creation statement `cursor.execute(queryCrashRecordTable)`

Then we use INSERT query to insert all the records showed by the following function:

```
def insertIntoTable(currentRow):
    # Insertion query
    queryInsert = "INSERT INTO `crashrecords`(`idCrashRecord`,
`crashDate`, `crashTime`, `borough`, `zipCode`, `latitude`,
`longitude`, `location`, `onStreetName`, `crossStreetName`,
`offStrretName`, `numPersonInjured`, `numPersonKilled`,
`numPredestrianInjured`, `numPredestrianKilled`,
`numCyclistInjured`, `numCyclistKilled`, `numMotoristInjured`,
`numMotoristKilled`, `contributingFactorVehicle1`,
`contributingFactorVehicle2`, `contributingFactorVehicle3`,
`contributingFactorVehicle4`, `contributingFactorVehicle5`,
`collisionID`, `vehicleTypeCode1`, `vehicleTypeCode2`,
`vehicleTypeCode3`, `vehicleTypeCode4`, `vehicleTypeCode5`) VALUES (
" + currentRow  + ")"
```

Figure 4: Python function inserting each row of data from the CSV to the MySQL collision table

The program took approximately one hour to copy the row of data and inserting them in the collision MYSQL table, as the dataset consists of 2.5 million of rows.

Now that the database is ready, PHP and MySQL will allow us to connect the database to the web interface. The development environment used is ATOM [14], specified in web development such as: HTML, CSS, PHP, and JavaScript.

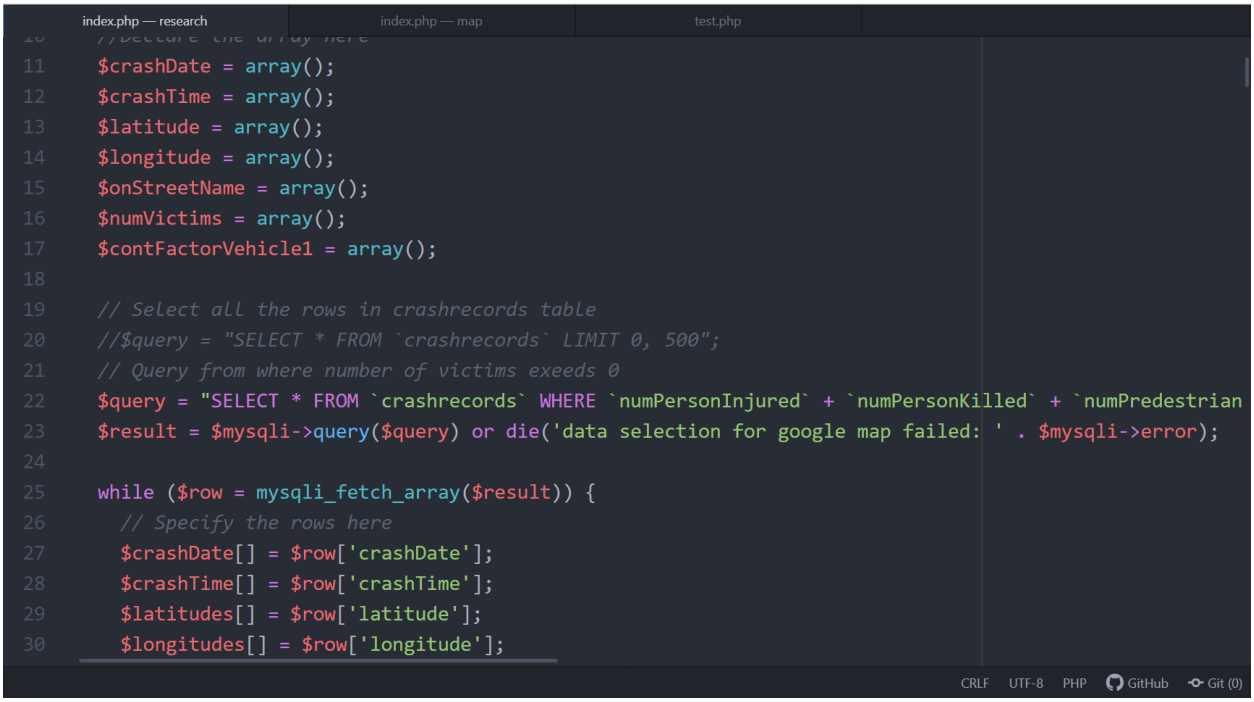

Figure 5: PHP script connecting the DB to the web page in ATOM.

The goal is to retrieve and display the locations where a collision recently occurred including the number of casualties, date, time, and reason behind each collision.

4.3 Google Maps API

On the other hand, the integration of Google Maps platform consisted of getting an API key from this website: developers.google.com then associate it with this project. The following shows the format required to include the API key include displaying a ready Google Maps template for our project.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Map</title>
    <meta name="viewport" content="initial-scale=1.0">
    <meta charset="utf-8">
    <style>
      /* Always set the map height explicitly to define the size of the div
       * element that contains the map. */
      #map {
        height: 100%;
      }
      /* Optional: Makes the sample page fill the window. */
      html, body {
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>
  </head>
  <body>
    <div id="map"></div>
    <script>
      var map;
      function initMap() {
        map = new google.maps.Map(document.getElementById('map'), {
          center: {lat: -34.397, lng: 150.644},
          zoom: 8
        });
      }
    </script>
    <script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"
    async defer></script>
  </body>
</html>
```

Figure 6: Google Maps JavaScript and API key integration

The API key is in a meta alpha-numerous formats and is provided by www.developers.google.com website. It is generally used to identify this project, which can later be used for mobile implementation on Android or IOS devices. Although the API key is not free, Google gives a 12-month free trial for this key.

5. Results

The combination of the Google Maps integration and the PHP script to retrieve data from the collision table led to a data visualization of recent collisions.

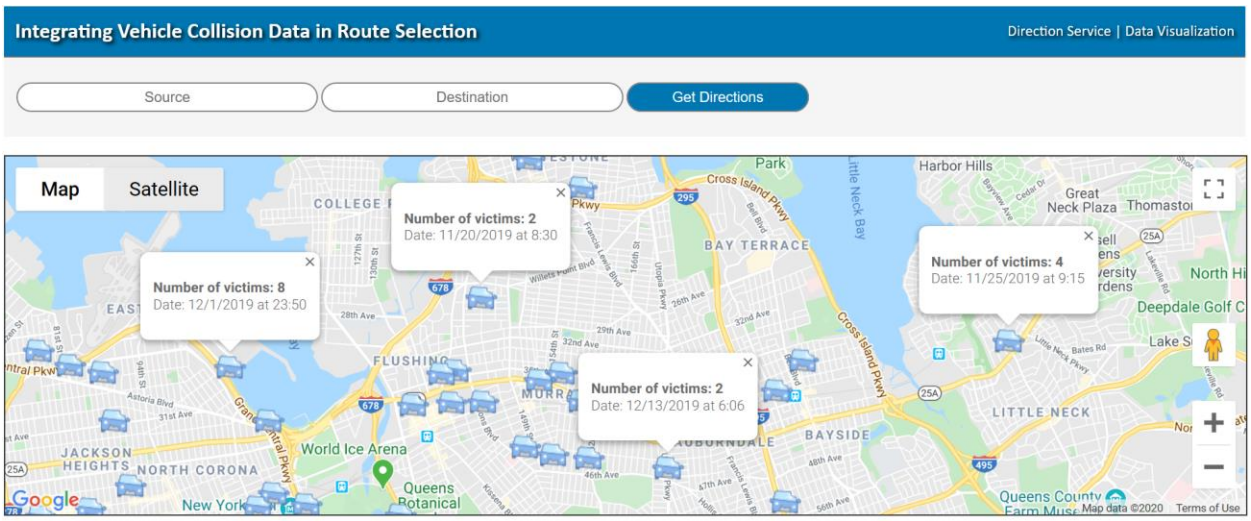

Figure 7: Visualization of vehicle collision from the input data.

In figure 6, some of the collisions can be visualized along with the number of victims including the date and time they occurred. Based on their exact location (latitude and longitude), we retrieved and plot them on Google Maps. Along the way of this research, the road trip planning is also displayed using the Google maps Directions API. This API consist of searching directions for several transportation modes, including driving as well.

```
function initMap() {
  var directionsService = new google.maps.DirectionsService();
  var directionsRenderer = new google.maps.DirectionsRenderer();
  var chicago = new google.maps.LatLng(41.850033, -87.6500523);
  var mapOptions = {
    zoom:7,
    center: chicago
  }
  var map = new google.maps.Map(document.getElementById('map'), mapOptions);
  directionsRenderer.setMap(map);
}

function calcRoute() {
  var start = document.getElementById('start').value;
  var end = document.getElementById('end').value;
  var request = {
    origin: start,
    destination: end,
    travelMode: 'DRIVING'
  };
  directionsService.route(request, function(result, status) {
    if (status == 'OK') {
      directionsRenderer.setDirections(result);
    }
  });
}
```

Figure 8: Integration of Direction API for road trip display

In figure 7, the function `calRoute()` is responsible of specifying the origin and destination of the trip, including the travel mode of the platform user. Then, the `directionsService.route()` checks and display the result into the map using the set directions method.

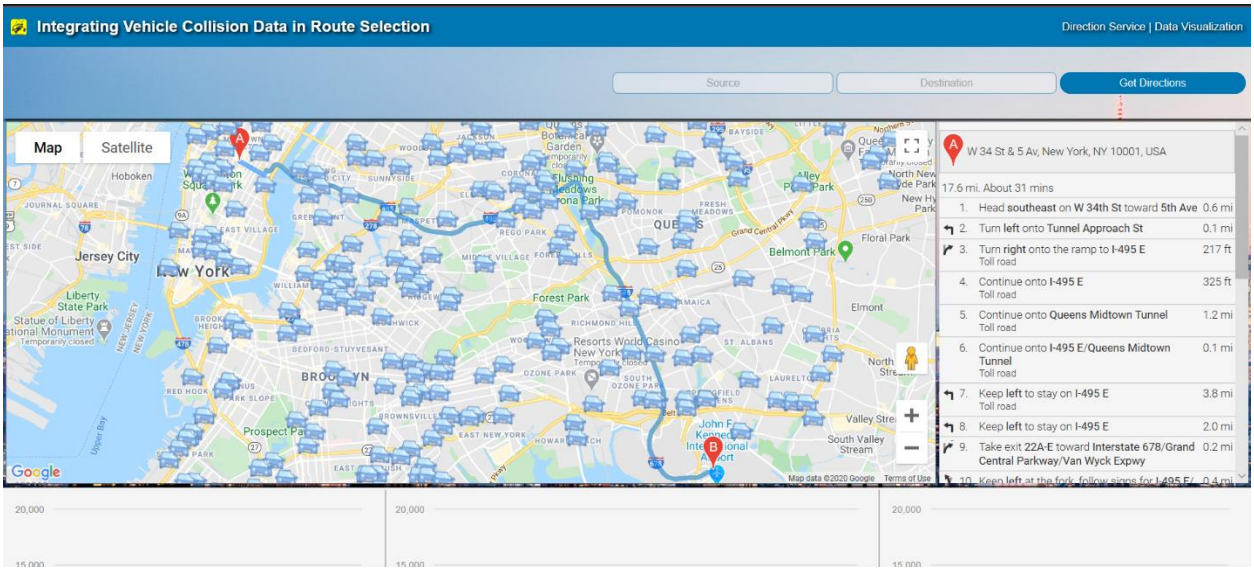

Figure 9: Display of collisions in a route trip location

From the route plan shown in figure 8, we can collect and display the steps and turns required until the user gets to his destination. The above picture shows the steps on the right side of this web page.

6. Conclusion

In conclusion, this research led us to develop this platform capable of integrating collision data into our trip planning. Several navigation systems exist nowadays but most of them focus on the distance, the speed, and the traffic size. This project focuses specifically on the aspect of integrating and visualizing traffic collisions on a route, to make the driver aware of the surroundings and thus reduce the risk of collisions. As GPS importance in our daily commute increases rapidly, thanks to the advancing of the technology. Not only this platform allows us to track recent collision according to our driving commute but it can also be used to study the type of collision that occur in specific areas, number of collisions that occurred as well on each road, and many more.

7. Future Work

Now, the platform is ready and performing two main functions: the data visualization for users to track recent collisions and the road trip planning service, along with the steps and turns required to get to destination safely. For future work, survey questionnaires will be conducted to research on whether and how much the risk-level affects when deciding on a route selection. The data analysis performed by this platform will also be focused on, such as: counting and summarizing the collisions displayed along the user trip to determine the frequent type of crash involved on each road of interest, to provide a summary of the surroundings. Based on this summarized information on the level or accidents on the routes of interest, users will be able to incorporate this when conducting route planning and empower the user to select the safest routes to get to their destinations.

8. Acknowledgments

This work is supported by the Louis Stokes Alliance for Minority Participation program (LSAMP).

9. References

1. Erel Avineri and Joseph N. Prashker. The impact of travel time information on travelers' learning under uncertainty. *Transportation*, 33(4):393–408, 2006.
2. D. Kwak, R. Liu, D. Kim, B. Nath and L. Iftode, "Seeing Is Believing: Sharing Real-Time Visual Traffic Information via Vehicular Clouds," in IEEE Access, vol. 4, pp. 3617-3631, 2016.
3. D. Kwak, D. Kim, R. Liu, L. Iftode and B. Nath, "Tweeting traffic image reports on the road," 6th International Conference on Mobile Computing, Applications and Services, Austin, TX, 2014, pp. 40-48.
4. D. Kwak, D. Kim, R. Liu, B. Nath and L. Iftode, "DoppelDriver: Counterfactual actual travel times for alternative routes," 2015 IEEE International Conference on Pervasive Computing and Communications (PerCom), St. Louis, MO, 2015, pp. 178-185, doi: 10.1109/PERCOM.2015.7146525.
5. NYC Open Data: https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95
6. Visualization Charts: https://data.cityofnewyork.us/d/h9gi-nx95/visualization
7. Google Maps APIs: https://developers.google.com/maps/documentation
8. Maps JavaScript API: https://developers.google.com/maps/documentation/javascript/tutorial
9. Direction API: https://developers.google.com/maps/documentation/directions/start
10. Places API: https://developers.google.com/places/web-service/intro
11. Geolocation API: https://developers.google.com/maps/documentation/geolocation/intro
12. MySQL: https://dev.mysql.com/doc/
13. PyCharm: The Python IDE for Professional Developers: https://www.jetbrains.com/pycharm/
14. ATOM: The Web development IDE for Professional Developers: https://atom.io/docs.