**ParkZone Web Application**

By:

Ramy Atalla, Michael Medina, and Mark Lindo

City University of New York

August 14, 2016

A problem many people in New York City have is they struggle to find a parking spot in busy areas. This poses an interesting problem as it leads to congestion, increasing the volume of cars on the road, and wastes time on the driver’s part in addition to causing unwanted stress. ParkZone is an application that, by way of web scraping, displays all parking garages and thier relevant information (rates, times, etc.) on a visual and easy to navigate map. Through this app, drivers will be able to find parking spots much easier, save themselves, time, hassle, as well as potentially cutting down on traffic and congestion at peak hours.

New York City is a condensed and active hub of many thing, the least of which are business both large and small, commuters going in and out of the city, and regular people, traveling all around looking for something to do. How do these people get to and fro? Public transportation and private transportation, namely cars. That being said, there is one issue that almost anyone who has navigated the city by car has faced. Upon arriving at their destination, they cannot find a place for them to park. Take a family from out of state. Their first time in New York City is surely a wondrous occasion and knowing all of the tourism site in Manhattan, they choose to spend a day there. Not knowing the seemingly complex MTA system and all of its ins and outs, they elect to drive into Manhattan. They go to Columbus Circle, relatively near to not only Central Park, but also 5th Avenue. However, before they can begin their day, they realize finding parking in this busy area seems almost impossible if unassisted. Take this family and multiply it by hundreds. This is the situation in many parts of the City every day, but what can we do about it. ParkZone hopes to address the problem of this family from out of state as well as all other people seeking to park in New York City. ParkZone hopes to

As stated before, the issue we are faced with is that every day, hundreds upon thousands of people struggle to find parking in New York City. There are several causes, some of them are addressable, while others not so much. One reason is that people do not know where they can or can’t park. The nature of the city of New York is that many places to park have restrictions as when when you can and can’t park. A spot on the side of the road may be a parking spot one day, but may not be another day. This can cause confusion for many people who are not already familiar with the street layouts and restrictions. Tied to this is that people are not aware of public parking locations such as lots, and garages. Again, this is something only people familiar with the streets would have knowledge of. Another problem is that people don’t want to pay for parking. Most people in New York City prefer to save money where they can and trying to find a free parking spot on the side of the road is one of the easier ways to save money. Paying for parking (say in a garage) is generally simpler and guarantees you parking but because of the often perceived high prices of garages, people try to avoid them. Tying into these issues is the problem of distance from your destination. Many people would like to find a parking spot that is as close to their destination as possible Even if there is a parking spot/garage, that is a bit far from their destination, many would generally not take it, in hopes that a parking spot closer to their destination is available. All of these amount to an issue many face every day and ParkZone hopes to address that.

ParkZone is a application that will help to simplify the process of finding a parking spot in New York City. The application is fairly simple in usage. The user will first navigate to the website, http://www.parkzone.us. Then they will arrive at the home page and map interface. From their, they can enter a location into the search bar and the application will then pull up all of the garages in their area, plot them on the map, as well as show them the prices, times, accessibility, phone number, nearby attractions, and a photo of the garage/lot. In addition, they can use a built in function to take their current location and plot that onto the map. The user can also, by way of a function, plot a course from their current location to the location of a garage of their choosing. The map interface also works like a normal map interface with controls to allow the user to zoom in and out, rotate around, and such. The user will also be able to toggle various elements of the map on or off such as the garages, parking signs, buildings, etc. Other features they can view (by way of map toggling) are areas where street parking is currently available as well as bicycle racks if they need a place to park their bicycle instead of a car. Finally, the user will be able to look at live feeds from several live traffic cameras that are available to the public in order to gauge not only traffic conditions in an area, but also potentially find parking spots on the road through these cameras. These are all fairly straightforward ideas. In short, ParkZone is an application that will draw data from a database and present it to the user in simple, visual, and interactive way.

Applications similar to this have existed and exist today, however, most if not all of these other applications do not present the information in a visual way and that is what sets ParkZone apart from the rest of them. Many other similar applications will, in the case of those provided by the garage/lot company, only gives information on their own garages/lots, while other applications that aggregate parking garage/lot data generally just list off all of the data they gathered. For the few that do offer a map interface, the maps are usually more basic and don’t offer much functionality outside of just viewing the data that has been plotted. ParkZone offers more functionality in that it combines many of these elements that are present in similar applications into one unified interface that is simple to use, easy to navigate and aesthetically pleasing.

ParkZone works by drawing data from a static database and displaying that information on a map interface. The database is divided into two main parts, the parking garages and the parking signs. Both sets of data were compiled via web scraping, the process of manually going to specific websites to extract data for some other use. The parking garages were gathered from various websites, included but not limited to the company websites of various parking companies like ICON, Edison FastPark, and SP Plus Parking Systems. These websites provided the address, phone number(s), accessibility, rates, and nearby attractions of the various garages and in addition to that, most also provided a photo of the garage in question. This data was taken and placed into a comma separated value (.csv) file that organizes the data and makes it simple to access. To properly plot the garages on a map, the geo coordinates are necessary. The addresses gathered earlier serve dual purpose here. The first is to provide the end user with an address they could navigate to and easily understand. The second is that the address provided a way to get exact geo coordinates of the garages. Using NASA’s online tool, all the garage addresses were manually entered into the tool and converted to exact longitude and latitude coordinates which were then added to the .csv. All of this is what comprises the database for the parking garages.

The parking signs were gathered in a different way. As the exact geo coordinates of each parking sign is not readily available to the public, a combination of tools were used to get the coordinates of each sign. First, using the online map provided by the New York City Department of Transportation, a map of all parking signs in New York City was obtained. To obtain the geo coordinates, the online website “geojson.io” was used. Geojson.io is a simple tool that can be used for viewing, sharing, and creating maps. The tool is based on the GeoJSON data format and is fully compatible. Geojson.io allows a user to plot points onto a map and returns to them various files containing the data of each plotted point. First, using very fine approximation, each sign plotted onto the Department of Transportation map was transferred to the geojson.io map interface. From there, the data of each sign (what each sign says specifically) was then added as a data field to each point on the geojson.io map interface. After all the points had finally been plotted, geojson.io has functionality to export all the plotted points as a .csv file. This .csv contained all the necessary information on the parking signs to be plotted onto ParkZone’s map interface.

The both of these databases are hosted by way of Django and their database creation tools. Using those, the database was imported and accessed using Django’s standard python interface. This makes hosting the databases as well as editing them fairly simple as all that is required to add or remove from the map interface is just modify the proper .csv files.

It should be noted that the bicycle rack data was not collected into a database in the aforementioned way. This data was available through public means and is drawn into ParkZone by pulling from outside sources. As such, the methods that apply to displaying the information on parking signs as well as parking garages does not apply for the bicycle racks.

The traffic camera feature of Parkzone required gathering information about The Department of Transportation's traffic cameras. This gathering of information was limited to the borough of Manhattan, although there are cameras located throughout the five boroughs of New York. The decision to limit the traffic cameras to the borough of Manhattan happened because of the method that the data was gathered. The data was gathered by performing by performing a manual webcrawl of the Department of Transportation's traffic camera website. This meant we gathered the data from the website and stored it in a .csv file to convert to a database table which will then be used to display the information to the user. The data of that was gathered was the URL of the live video feed from the Department of Transportation’s website. Also the name of the location of the camera the reason that the name of the camera was taken as well is because it would help get geo coordinates which was needed for the mapping of the location of the cameras. Then in order to get the geo coordinates there was a conversion that needed to be made. This involved taking the names of the locations of the cameras and converting them to longitude and latitude. This was done with the help of the Google Maps website but it was done by copying and pasting so it took longer than if it were done in a programmable way. This process also contributed to the limiting of cameras to only the borough of Manhattan.

In making the database the first step was creating a .csv file which would hold the data of the cameras that were collected from the Department of Transportation website. The way that the .csv file was made is that for each column would represent a feature and a row would represents an instance of a specific camera. Features for each camera were an ID which was an integer representing a unique identifier which is important in a database, longitude and latitude which were both rational numbers representing longitude and latitude this was important placing markers where cameras are located on the map. The final was the URL of the video feed of that camera. After the data was collected the .csv file was imported into the database which contained other tables for other functions of ParkZone using a database browser which is a tool that allowed us to connect to our database so we could browse and modify data on the database. Then after the .csv file is inserted to the database Django is able to access the data.

Since Django was used for the back-end then there were certain files that had to be included in our application in order to make our application work. In particular the models.py file had to be included. This file is the single, defining source of information about the data in the database, so this was written in a way that was able to contain the essential fields and behaviors of the data. For this camera feature a class was made and it had five features which were the id which is an integer that is unique to each instance, But the uniqueness is handled SQLite database, the place This is a string that contains the name of the location that was given by the Department of Transportation's traffic camera website, the Latitude which is a rational number representing the latitude of the location of the camera, longitude which is a rational number representing the longitude of the location of the camera, and the URL which is a string that holds the URL of the live video feed of the camera. This was most of the backend programming that was done for the traffic camera feature, the reason for this is that the functionality of the program was on the front-end.

Various amounts of tools and languages were used to make this feature possible. Sqlite was used as the database language, python was used for the back-end functionality that interacts with the database, HTML, CSS and JavaScript were used for the front-end functionality. HTML was used to display the information to the user CSS was used to format the information was displayed by way of the design of the website, and JavaScript was the functionality of the site. The cameras are displayed to the user through an interactive web page. What makes it interactive is the icons on the web page the changes the current website depending on which icon or button is pressed. The icon that allows the camera information to be displayed is a combination of HTML and CSS. One the one of the icons or buttons are pressed there is a JavaScript function that is called to display the information, this is what makes this web page dynamic and is also part of the user interface. Most of the front-end programming was done in the index.html file. This contained the majority of the HTML, CSS and JavaScript was required for the traffic cameras feature, it also had some of the back-end through something that in Django is called a template which contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

This dynamic part is written in Python and for this traffic camera feature it takes the data that is in the database and makes it accessible to the front-end then a JavaScript function inserts it into the OSMbuildings layer and marks the locations that were in the database by placing yellow ParkZone markers on the map. The way that this Happened the application is that the user clicks on a gray camera button next to the search bar once the camera button is clicked the JavaScript function is called after that the markers are placed on the map. The markers that are placed represent 153 traffic cameras that were collected these markers are shown as 3D objects. in order to make these markers clickable there is another JavaScript function that was made, this function is an onclick listener The way this works is any time the map is clicked the function will be called inside of the function it checks the ID of the object that has been clicked there is an if statement inside of the function if the item that has been click is I yellow parkzone marker then the live video corresponding to that location will be pulled up in a pop-up window. The window has the live video feed from the Department of Transportation website. From there the user is able to view the traffic camera video feed but this is another website, which has the Benefit of not slowing down the Parkzone webpage. Thee pop up browser window that opens is independent from the Parkzone web page so the user can continue to use other features in the Parkzone webpage but that does not included opening a new live camera feed window. If a new window is opened then the current window showing the video feed will close and the new window will be opened, taking the place of the old video feed.

After the databases were built and access was done through Django, the map interface was designed. Mapbox was used for the map interface. Mapbox is a platform that provides various map interfaces of varying styles for use. Mapbox provides several other features to use in conjunction with their maps and ParkZone does utilize some of them such as search and geocoding as well as directions (the directions are plotted specifically with MapQuest but then utilize Turf in OSMBuildings to plot). In addition to that, Mapbox is compatible with several other libraries and API’s to help enhance its functionality. One such that were utilized in ParkZone were Turf which is a Javascript library that is used to draw, what became in ParkZone, lines that delineate routes on the map for ParkZone’s “Get Directions” functionality. Mapbox is one of the core components of ParkZone and the foundation for all additional functionality on top of it.

OSMBuildings is the other core component that allows ParkZone to function in the way that it does. OSMBuildings is a Javascript library that allows one to build geometry on top of 2D and 3D maps. This is used to do several things in ParkZone. The first is building the 3D models on the map interface. The buildings do not serve a purpose outside of giving the user the ability to locate buildings if they can discern their shape. This is largely for user situational reference more so than anything. The second and more important function that OSMBuildings is serving in ParkZone is the plotting of markers that serve as indicators for the various elements on the map. When using ParkZone, parking garages are indicated by a small red ParkZone “P”. The user’s location is indicated by a small blue ParkZone “P”. The bicycle racks are indicated by a small yellow ParkZone “P”. The parking signs are indicated with miniature parking signs. All of these markers were 3D modeled separately and using OSMBuildings’ tools, were then placed onto the map interface for use with the application.

To touch on a few other tools used, a few scripts were used to add various animations to the site from a variety of places, the about us page for example or some of the animations that are used on the map interface. Also as stated before, Turf was used to draw the routes for the direction feature on the map interface. The main webpage was built and modified off of a preexisting template.

As stated before, ParkZone was designed to be a simple to use. Upon arriving at the website, the user will be presented with a map interface, a header bar with a few clickable links labeled “How It Work” and “About Us”, and a few map controls on the right side of the page. The center of the page is occupied by a search bar and a “my location” button. On the header bar, the ParkZone logo at the top left is a clickable link that redirects the user back to the main page. The “About Us’ link redirects the user to a page where they are shown a short biography of each of the developers behind ParkZone as well as a photo of them. In addition, at the bottom of the page is an icon that redirects the user to the GitHub repository for ParkZone. When on the “About Us” page, the how it works button becomes a “Home” button redirecting the user back to the home page. On the home page, clicking the”How it Works” button at any time grays out the map interface and superimposes on top of it a series of labels detailing what each element on the screen at the time represents and what it does. The map controls are located at the top right of the map interface. These provide basic controls for navigating the map such as zoom, rotation, as well as angle of elevation as the map is a 3D rendered map. Below the controls is a pull out menu that toggles on or off various elements of the map. With this the user can choose whether or not they want certain elements to load such as parking garages, parking signs, or even the buildings that are rendered.

Next to the search bar is a “My Location” button. Clicking this at any time will utilize the device's’ GPS functionality to determine the user’s current location and plot a marker on the map where the user is. Finally at the center of the page is the search bar. When the user enters an address, ParkZone, access the database and plots markers of all parking garages, bicycle racks and parking signs in the area. In addition to this, the search bar moves itself to the top left corner of the screen and a list of those garages is displayed in a scrolling list down the left side of the screen. This list details the parking garage’s name, phone numbers, prices, operation hours, nearby attractions, accessibility and service features, as well as a photo of the garage. Clicking any of these list entries causes the map to zoom in on the garage in question and highlights it green. Finally, each garage in the list has a “Get Directions” button. Clicking this causes ParkZone to determine the user’s current location and then plots a course, using the aforementioned Turf, from the user’s current location to the garage in question. It also returns to the user directions in a separate window. The interface is clean and simple and due to this, most users should not have difficulty in using ParkZone.

ParkZone hopes to change the way people find places to park. Our goal is to streamline the experience and make it simple for people familiar and unfamiliar with the area to be able to find the best garage or lot for them to park in. Other applications exist in a similar spectrum but few if any brings together so many elements into a simple to user experience. ParkZone will continue to grow in the future with more features added the more they seem to become necessary. The team behind ParkZone will continue to work and help this application grow to not only become the best application of its kind but to also push the market forward to developing even more applications similar and to push other applications that already exists to new heights and help them develop to become even better, all for the sake of helping everyone who uses them have a simple, easy, and streamlined parking experience.