An Intelligent Parking Assistant

Nishok Narasimha-Mohanasamy
Computer Science Department
Montclair State University
Montclair, NJ 07043, USA

and

John Jenq
Computer Science Department
Montclair State University
Montclair, NJ 07043, USA

ABSTRACT

In big cities, parking is a very challenging problem to most people especially for visitors who are new to these cities. Most people today will not get lost in a big city due to the widely use of smart devices. But to find a nearby parking spot which is close to the place you want to visit can be a hard task. Not just smart devices like smart phone become popular, we can expect the autonomous vehicles will become popular a norm in the very near future, an internet based parking system is now on demand. There are parking assistance available now on internet. Most of these systems ask user to enter information such as where you are heading to, when you plan to arrive, how to charge you and then make reservation for you, etc. The challenge is how we build an intelligent system which is compact, fast responsive, and smart enough to help user to find and route user (or auto-driven car) to a nearby parking place whether it is a garage, surface lot or on the street. In this report, we propose an intelligent parking assistance system as a component of the future smart cities. We use MEAN stack to implement a prototype parking system. MEAN is JavaScript based technology which stands for MongoDB, Express, AngularJS, and NodeJS. We develop such a parking assist application purely using web based technologies such as HTML, CSS, JavaScript and its open sourced frameworks such as node.js, bootstrap, jQuery, jQuery UI etc. Since that real time aerial image are not available so we use simulation to find parking information on open surface lots and street parking.

1. INTRODUCTION

Parking is a big problem in most of the places, especially in a big city like New York, Chicago, and Boston. Now that smart devices like smart phone become so popular, and the autonomous vehicles will become a norm in the very near future, an internet based parking system is now on demand. There are parking assistance available now on internet for example [1][2]. Most of these systems ask user to enter information such as where you are heading to, when you plan to arrive, how to charge you and then make reservation for you, etc. The challenge is how we build an intelligent system which is smart enough to help user to find and route user (or auto-driven car) to a nearby parking place whether it is a garage, surface lot or on the street. Regardless you have to pay or not. We understand there are issues the society need to resolve in order to support such a fully automatic system. One of the benefit of such a system is to reduce the car emission which can protect our environment and the other benefit is save time for people. It is easier to achieve such a system by inviting garage owners to participate in such a system. For street parking, we need the government to allow access to all city street...
situation in real time, or if big company such as Google can provide real time data. In this report, we use MEAN stack[3][7][12] to implement a prototype parking system. MEAN stands for MongoDB[8], Express [9], AngularJS [10], nodeJS[11]. We develop such a parking assist application purely using web based technologies such as HTML, CSS, Javascript and its open sourced frameworks such as node.js, bootstrap, jQuery, jQuery UI etc. Since that real time aerial image are not available so we use simulation to find parking information on open surface lots and street parking. We expect in the future smart cities, these information will become available and these aerial images can be processed and information be retrieved.

For garage parking, this system can be implemented easily. Because of the following reasons: (a) most of the parking garage are e-controlled (b) almost all the parking garages use video monitor to monitor their garages (c) software for image processing and pattern recognition are widely available (d) the availability of indoor GPS. In our system, we use node.js as our web server. The node.js is an event-driven I/O server-side JavaScript environment based on V8. Companies such as Walmart [4], ebay[5] to name a few promoting using node.js. The Javascript callback feature make node.js popular. Walmart once claimed to serve customers on black Friday which greatly reduce the cpu time by using node.js technology [6]. We use Mongodb as our database in which we put some makeup data on it which represent parking space availability data there so that we can retrieve and return to user app.

The popularity of GPS make one’s location readily available even when one is on the move, most of the smart phone apps use this information to provide user with services. Google provide many useful APIs for developer so that one can get distance, compute travel time and provide routes and alternate routes etc. Since these APIs are mostly provided as content delivery network (CDN), it provides high availability and high performance to the end users. The core components of our report consists of Google APIs: Maps Javascript, Places Web services, Direction services, and Distance matrix services. We also used MooTools[13] classes: Park regions, and Markers. Section 2 of this report gives the system design and implementation. Section 3 contains simulation results. We conclude the report in section 4.

2. SYSTEM DESIGN AND ARCHITECTURE

This system components and architecture is shown in figure 2.1. It is called Parkit.

**Figure 2.1. ParkIt architecture**

The main page of the web server is called index.html. This page is located in the view folder (park-it\views\index.html). There is a Javascript which loads the google map on the body and calls the callback "initMap" after loading the map. <script src="https://maps.googleapis.com/maps/api/js?key=<KEY>&signed_in=true&libraries=places &callback=initMap" async defer></script>.

The initMap initializes ParkItPage, which is a utility and base class available throughout the lifecycle of one session. This class also holds methods such as openInfoWindow (google's info box for markers), calculateAndDisplayRoute (googles directionsDisplay and directionsService API), displayParking (google getPlaceDetails Ajax request), displayIndoorMap (displays indoor map). After initializing ParkItPage, initMap adds listeners such as zoom_changed, place_changed et cetera then starts initializing ParkItRegion class, which fires getRegionData. The callback for getRegionData creates the parking place markers. The place markers are
created using a class Marker, which has an important timer functions getAvailability, which gets the availability status of the particular marker. This timer function calls itself for every 8 minutes. Here 8 minutes is just a randomly set timer. It can be set differently if the system were implemented in a real world. When the autocomplete's address input changes with new location, the ParkItRegion will be re-initialized with a new position.

The main Javascript is called parkit.js which is located in the javascript folder (parkit/public/javascripts/parkit.js). This Javascript contains three important classes: ParkItPage, ParkItRegion, and Marker.

ParkItPage class, which is a standalone utility class. It contains static methods such as openInfoWindow, calculateAndDisplayRoute, et Cetera, is created only once during the entire lifecycle the web page.

The second class is ParkItRegion class, which is the main holder class created on every zoom changed or new location change. It holds the API getRegionData for fetching parking information using google places nearbysearch AJAX and the API (createMarker) for creating markers on maps.

The third class is Marker class. This class is a placeholder class created for every parking slot, holds all the necessary API for that specific slot such as getAvailability, setIcon et cetera.

The main server page is called index.js which is located in the routes folder (parkit/routes/index.js). This is the main router for the backend server which processes all the AJAX request from the front-end including google API AJAX calls. Google's Places API AJAX calls are re-directed to this route, then this route handles these requests with fail back.

This Javascript contains the following GET requests: getData, getAvailability, getAllPlaceIds, getAllAvailability, removePlace, addAvailability, insertParkers, updateAvailability, addPlaceToGoogle, deletePlaceToGoogle, getPlaceDetails

3. EXPERIMENTAL RESULTS

Figure 1 shows the user interface which show available parking lot. Figure 3.2 shows the detailed routing information.

Figure 3.1 Parking lot chosen

Figure 3.2 Routing Information

The detailed parking lot information can be displayed as in Figure 3.3 which shows a detailed parking lot information which user will park.

Figure 3.4 shows indoor parking. The data was from simulation. Here we assume the owner of the parking garage is participate in ParkIt business. The green spots show the available parking spots, while the red ones are taken. In a real system these information can be acquired from a pattern recognition subsystem. The data were stored in MongoDB.
example, the parking scheduling component can help the system to minimize the total and individual car routing time and reduce the car emission. We can embed priority into this system so that it can respond and accommodate emergency situation.

5. REFERENCES

[8] https://docs.mongodb.org/manual/
[10] https://docs.angularjs.org/guide/