Parking Oriented Travel Planner

Yu-Lin Chung, Ting-Yu Kang, Xiaofu Niu, Youyi Shi, Chuyun Sun, Qian Yu

ychung82, tkang49, xniu37, yshi357, csun71, qyu48

ABSTRACT

This is the final report for CSE 6242 project *Parking Oriented Travel Planner*. In this project, we designed and implemented a full-stack project web application that provides one-day travel route planning for customers.

Keywords: Route Planning, Recommendation, Full-stack

1 INTRODUCTION

More and more people like self-drive tours these days. However, how to make a tourism plan that uses less time and cost to visit more spots is an essential aspect that a tourist most cares about. Therefore, making an efficiency tourism schedule will be a new challenge for most people when they start to make a tour schedule. Based on users' travel preferences, our application will plan travel routes with parking lot information included.

There are so many tourism tour planning applications in the market, but none of them includes parking information. As more and more people choose self-drive tours, it will be a high demand for an application like this. Time-consuming attraction finding and parking lot searching can kill all the joys of traveling. Our application will take care of all those problems that would estimate traffic time taking walking from and to parking lot into consideration.

2 MOTIVATION

1. The first travel planning application combining location of tourism spots finding, travel route planning, and parking lots searching.
2. Self-design route planning algorithm to generate the most efficient route covering all spots user wants to visit.
3. Estimate travel time and include restaurant information at dining time.
4. Visualize the planned route with user friendly interface and which is easy to work with.

3 PROBLEM DEFINITION

In this project, we are attempting to solve these three questions:

1. What is the best strategy to integrate attraction data and parking lot data to provide travel route planning?
2. How can we further enhance the route planning experience (like providing restaurant options)?
3. How can we display rich information in a clear manner to provide users the best experience?

4 SURVEY

Although there are many pieces of research focused on estimated and shorten the travel time. (1) discussed several algorithms that generate the shortest path and proposed which can be fast enough to achieve real-time traffic updates. (2) used the distance and time proportion approach and the spatial and temporal moving average approach, (3) relies on the links or paths. (4) used the GPS-equipped taxi trajectory data to set up a tensor-based spatial-temporal model. (5) and (6) both mentioned computing traffic conditions through Spatio-Temporal Random Field from sensor reading and the possibility of lower sensor coverage by applying Gaussian Process Regression.

Nonetheless, (7) found out that drivers pick the fastest driving path for only 35% of the journey, which means there are other personal factors that affect route choices. For example, (8) suggested that tourists' movements are impacted by urban's spatial arrangement. (9), (10) and (11) proposed a highly personalized route planning system for individual users. (12) found that first-time visitor tends to explore more places. They didn't mention parking lot but for a car driver, it is an important factor to decide how would they move. Both (13) and (14) found out that there are certain clusters in the city that tourists tend to walk around them. (15) predict the travel time at the link / path level by the K-nearest neighbor algorithm. We pick 0.5 km as our radius as suggested by (16) and (17).

From (18) and (19) we learned that Google Map provides not only the address but also the characteristics of the address. It can help us to locate our targets in the geographic regions in map visualization. We try to get the travel time for the whole trip by using Google Map API, where we can collect real-time data.

5 PROPOSED METHOD

5.1 Why is our design better than the state of the art?

There are a lot of travel route planners be active in the market, such as Roadtrippers, GoogleMaps, Spotify and so on. Current applications only focus on point to point tourism planning depends on tourists preference. Different from those traditional applications, our route planner provide suggestions with a lot of travel request information, thus tourists will know what places they can visit when they are in a specific area.

Another advantage of our application is that we only do planning route for travel. Specially focus on parking information, restaurant recommendation, and attraction suggestions. So we will finally become a number one tourism application that people first come to mind during their travel time.

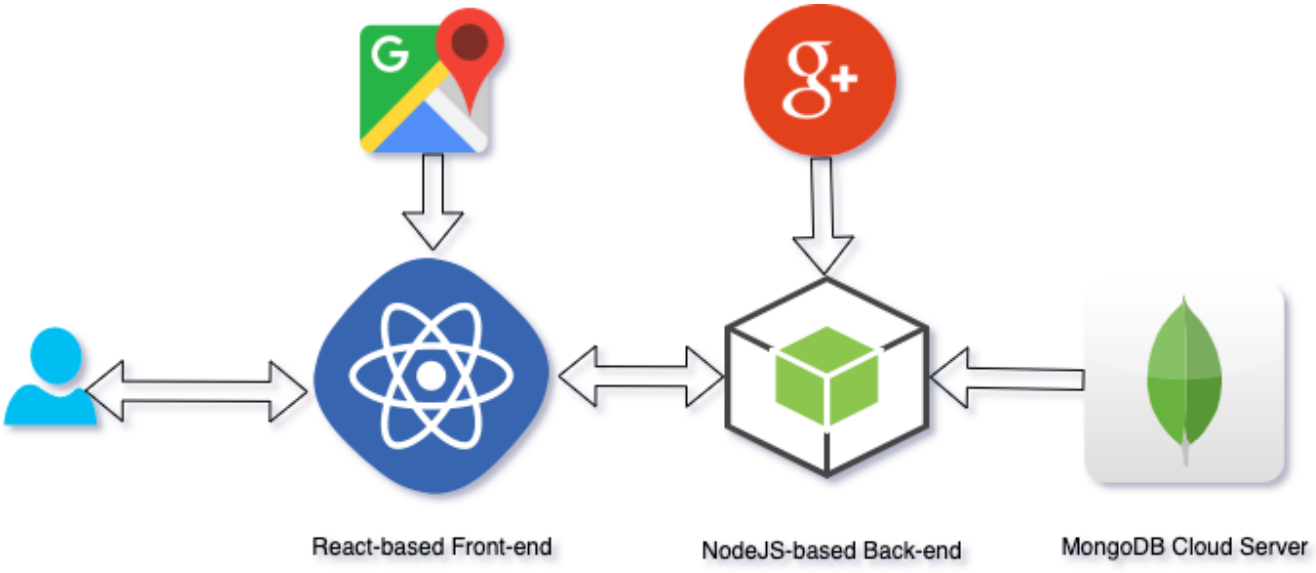

Figure 1. Application Data Flow

5.2 Database

To achieve the most comprehensive travel assistant application, we combine data from multiple platforms and bounded Los Angeles area by restricting the latitude between 34.3 and 33.7 and longitude between -118.15 and -118.67.

- We use **Yelp**(20) and **Google**(21) APIs to obtain **attractions** and **restaurants** in Los Angeles area. It gives us access to detailed information of attractions such as latitude and longitude, categories and subcategories, rating, price and other information. In the end, we have around **2,300** attractions in our table that are split into 4 categories and 18 subcategories. Yelp also provides information on location, price and rating of restaurants. We obtain about **4,400** restaurants in total for 4 types of cuisine. To evenly distribute attractions, we check the density of attractions and filter out some attractions where the density is too high. The result of this step is shown in the next section.
- Data of **parking lot** locations and prices can be obtained from **Here**(22), a parking mobility platform. The website enforce a limit that we can only get data for 20 parking lot thus we decided to use the location of each attraction as center and find nearby parking lot for each of them. In this case, we were also given the distance between the attraction and corresponding parking lots, which can then be used to ensure attractions and parking lots are within walking distance(1 mile). Originally, there are over 40,000 parking lots and we reduce the size down to around **24,000** after setting the distance limit to 1 mile and dropping all duplicate parking lots.

The last step is to setup database by importing our data to **MongoDB Atlas**(23) cloud server. Three collections are created for data of attractions, restaurants, and parking lots. For attractions and restaurants we also have a collection to map category name to category ID.

Parking lot database schema

ID	Name	Lat	Long	Address	Distance

Restaurant database schema

ID	Name	Lat	Long	Address	Phone	Rating	Price	URL	Cat	City	Zip code

Attraction database schema

ID	Name	Lat	Long	Address	Phone	Rating	Price	URL	Cat	Sub-cat	City	Zip code

Figure 2. Data Schema

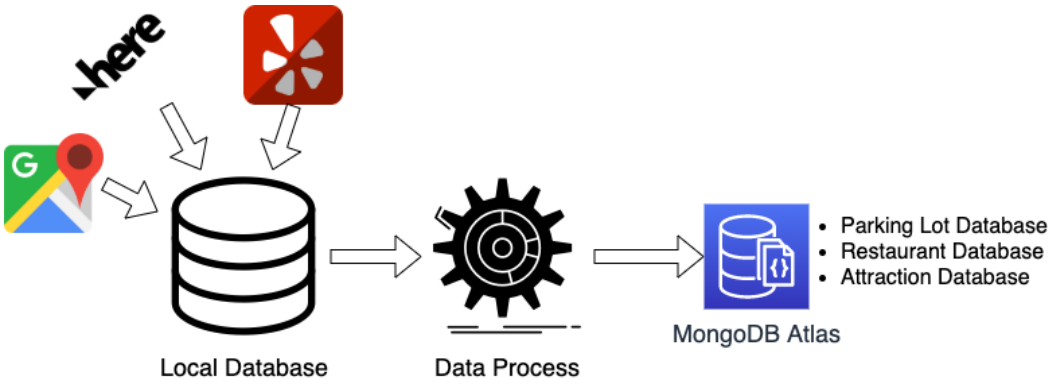

Figure 3. Data Preprocess

5.3 Front End

Our application try to give our client a best route of travel which includes attractions, restaurants and parking information. There are four pages to collect client preference and show the final route on the map. We use react to realize our front end interface. In order to provide a good user experience, we implement materials UI module to organize our webpage. For the main route planning component, we implement GoogleMap API to show our route algorithm. Specially, we highlight the attractions, restaurants and parking location with different shapes that user can clearly get their route. It is a really customize application which based on users' preference.

Page 1 - Client Preference Check Page

- Input: Client need to check attraction type on this page which they want to go.
- Output: All the checked attraction points will send to back end.

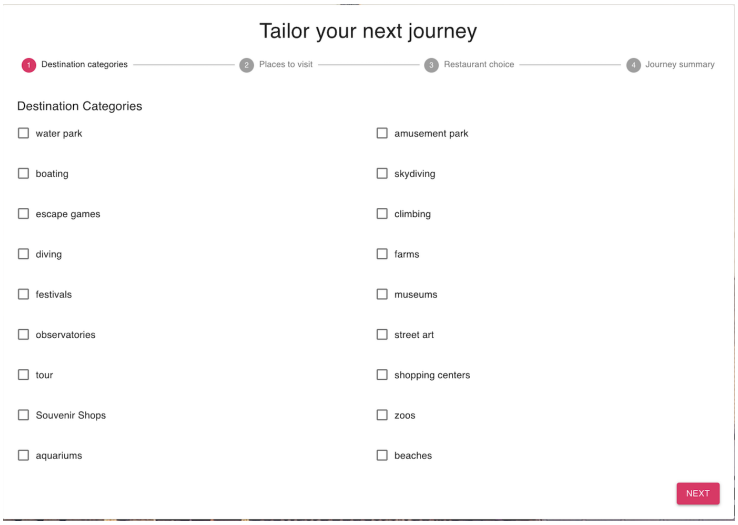

Figure 4. Front End Page 1

Page 2 - Client Attraction Check Page

- Input: Client need to choose a list of attractions they want to go and their stay time.
- Output: All the chosen attraction places and stay time will send back to back end.

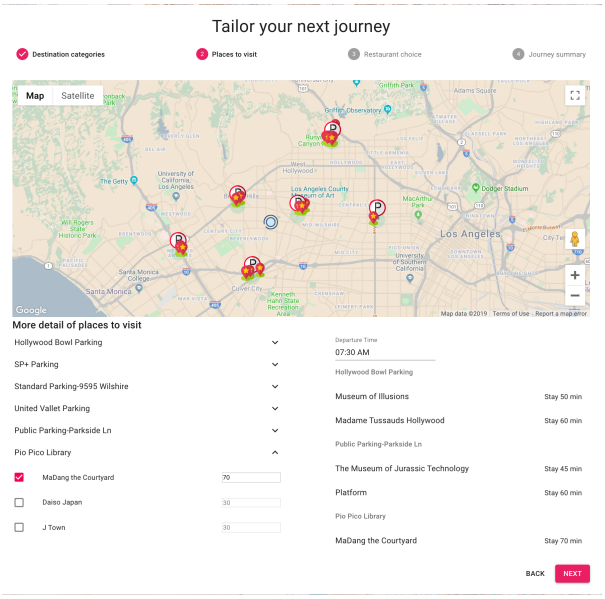

Figure 5. Front End Page 2

Page 3 - Restaurant preference Page

- Input: A list of recommended restaurants for client to choose. It has some many restaurants can choose, like Chinese food, India food, fast food and so on.
- Output: All restaurant clients want to go will send back to back end.

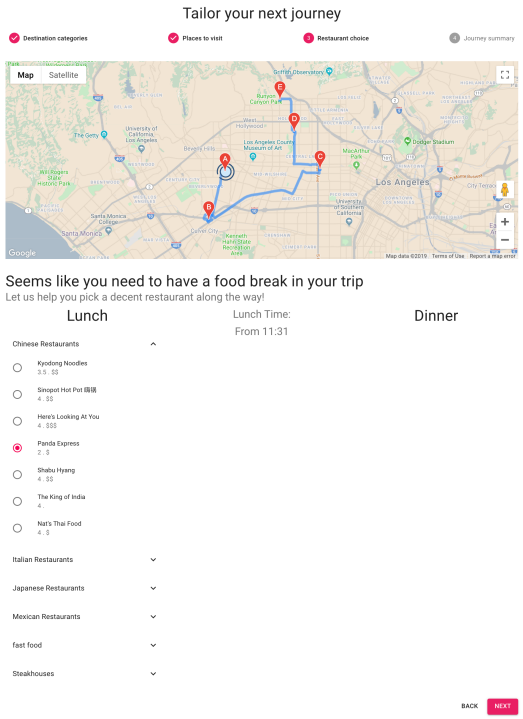

Figure 6. Front End Page 3

Page 4 - Route Planning Finalize Page

- Input: List of attractions and restaurant with visit order sending from back end
- Output: Map with finalize route recommendation for client.

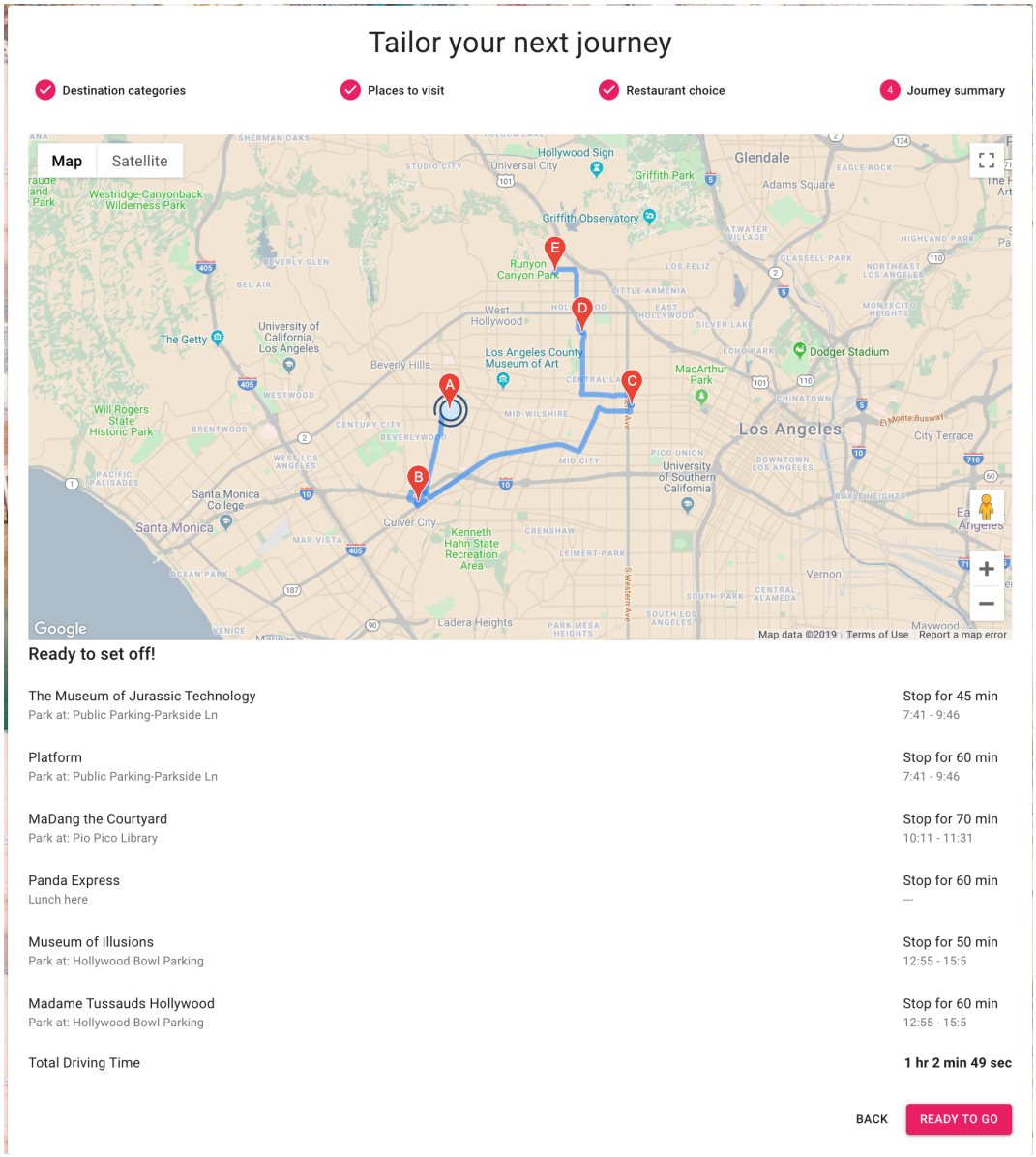

Figure 7. Front End Page 4

5.4 Back End

Similar to front end, our back-end is also divided into multiple steps. We display results of each step to users and collect input for the next step. The general diagram is shown in figure 8

Step 0 - Preprocess

In this step, we are not doing any computation. We just send all attraction categories and restaurant categories to frond-end for display.

Step 1 - Finding Attraction Groups

This step is the intermediate stage between page 1 and page 2 in front end. We take input as user's preferred attraction type and return a list of attraction groups. Each attraction group consists of a parking lot and some attractions within walking distance from the parking lot. The algorithm is defined as follow:

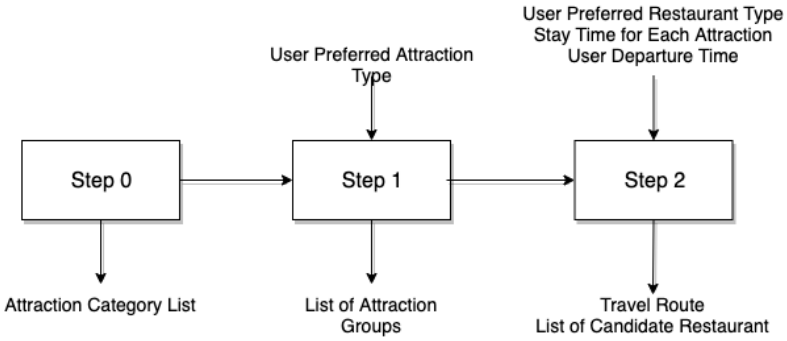

Figure 8. Back End Diagram

Algorithm 1: Finding attraction group

```
Fetch 500 parking lots within driving distance from current location
Fetch all attractions in user's preferred style
for each parking lot do
    for each attraction do
        if attraction is within walking distance from parking lot then
            Increase the score of parking lot by 1
        end
    end
end
Sort all parking lot by score
while number of attraction groups ⩽ 6 do
    if current parking lot satisfies all requirements then
        Add current parking lot and attractions around it as a new attraction group
    end
end
```

Step 2 - Route planning

Step 2 is the intermediate stage between page 2 and page 3. Besides, restaurant data which will be displayed on page 4 is also collected and sent to front end at this step. The input to this step is user's choices of attractions and expected time to spend at each attraction and departure time. We output a driving route among attraction groups and list of restaurants.

Algorithm 2: Route planning

```
Use DFS to find all permutations of all attraction groups, each permutation represents a route
for each route do
    Calculate total driving distance
end
Sort all routes by driving distance
Find the best route with shortest driving distance
Calculate driving time for each path in the route using Google Distance Matrix API
Estimate arrival and departure time for each attraction group
Find the path_lunch and path_dinner on which customers will have lunch and dinner
Draw an ellipse using path_lunch as long axis, find all restaurants within this ellipse
Repeate step 10 for lunch
```

6 EXPERIMENTS & EVALUATION

6.1 Database

We conducted two experiments for database. First one is to check if we have evenly distributed locations within the region. Second one is about the relative location of attractions and parking lots, which checks if there are parking lots within walking

distance of attractions.

The procedure of first experiment is to plot our attractions and see the density of data points. If the data point is too dense in one area, then we remove some of attractions. The figure 9 below demonstrates the results of our first experiment. From left to right, the three graphs show the distribution of attractions with subcategory zoos, the distribution of restaurants with subcategory Chinese restaurants, and the distribution of parking lots in Los Angeles. According to figure 9, our group have fairly even data points in Los Angeles regions.

For second experiment, we check for parking lot availability around our attractions. We complete this by checking the distance between attractions and parking lots and drop those with a distance that is greater than 1 mile(1600m).

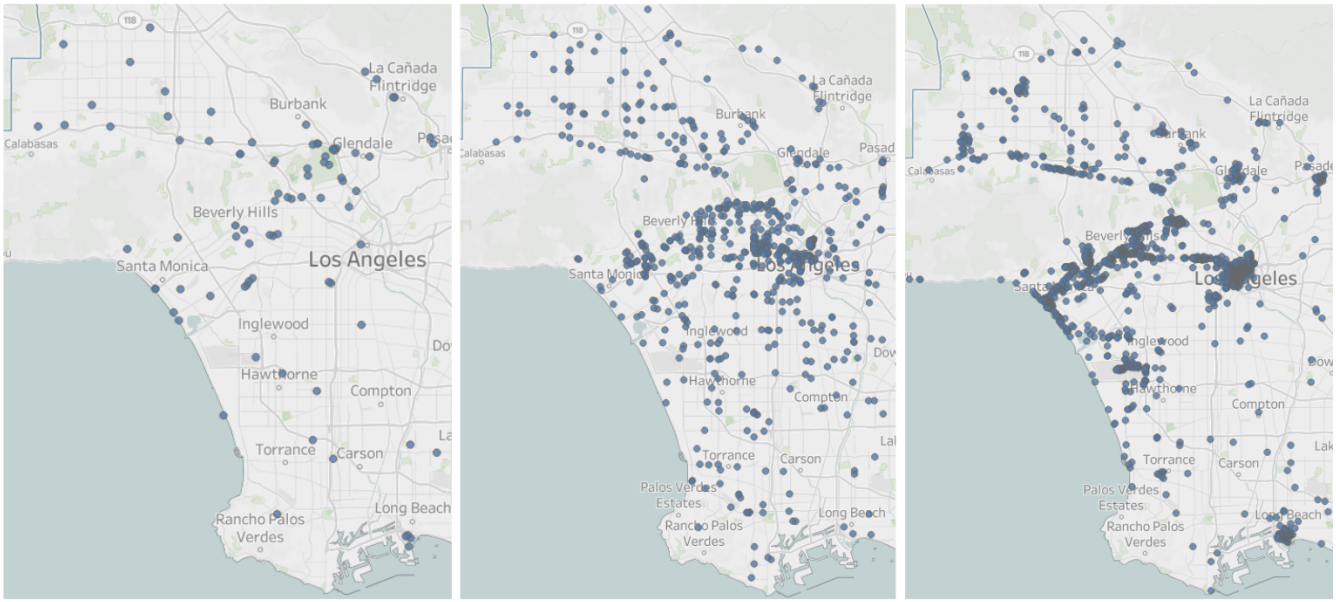

Figure 9. Result of database experiment 1

6.2 Frond End

Experiments

For the front end interface, we came up with two experiments. The first one was front end interface construction check experiment. We set up a fake test data which mimic the data sending from back end to test the connection and the layout for our front end. We tested with different number data that to test the web-page adjust layout. Also, we tested with different window sizes to make sure our application always satisfy users' preference and users' can get a clear route recommendation all the time. The second one was front end interface route check experiment. We connected to the back end and asked for different attractions and restaurants combination. Then we manually check the shortest route among those target location. Also, we looked at the map on our website to see the route display.

Evaluation

Based on our two types of experiments, we can see that our front end interface can handle different window sizes and different number of data provides on our web page. And our fancy web page design get a lot of encouragement from our tester. Also, our route display section give tester a direct and reliable route recommendation which become the greatest selling point people want to try our application. On the other hand, we collect a suggestion from our testers. They think our project has a really good idea and has a great user experience. But we only focus on LA right now. If our application can include more cities with route recommendation, it will be perfect. Our group will take this suggestion in consider, and write on our future work list.

6.3 Back End

The evaluation of back-end is divided into two parts: in the first part we verify the correctness and accuracy of our algorithm and in the second part we evaluate the time efficiency.

For the first part of evaluation, we tested step 1 algorithm on different inputs and counted the number of attraction groups found. The result is shown in table 1. We observed that the number of attraction groups varied according to user's current location and user's preferred attraction types. There exists situations in which no attractions were found based on user's current

location and attraction type, like the one shown in table below. No attraction of type 'boating' was found when starting from location 3. Location 3 located in a neighbourhood in Einco. This result still makes sense since there is no boating facility around Einco. To evaluate the accuracy of step 2 algorithm, we observed the output route of step 2. Some of the results are shown below. The fact that no crossover existed in route proved that our route planning algorithm is efficient.

Table 1. Number of attractions on different inputs. Style 1 = {zoos, shopping and beaches}, style 2 = {festivals, museums}, style 3 = {boating}

	style 1	style 2	style 3
location 1	22	40	11
location 2	27	67	4
location 3	9	3	0

Next we evaluated the efficiency of algorithm in terms of response time. Since our database has over 24,000 parking lots, the overhead of finding parking lots used to be very heavy. We solved this problem by limiting the number of parking lots to 500. The result is shown in figure 10 Similar to what we did in the first step, we test step 1 and step 2 algorithm on different inputs and record the average response time at front-end.

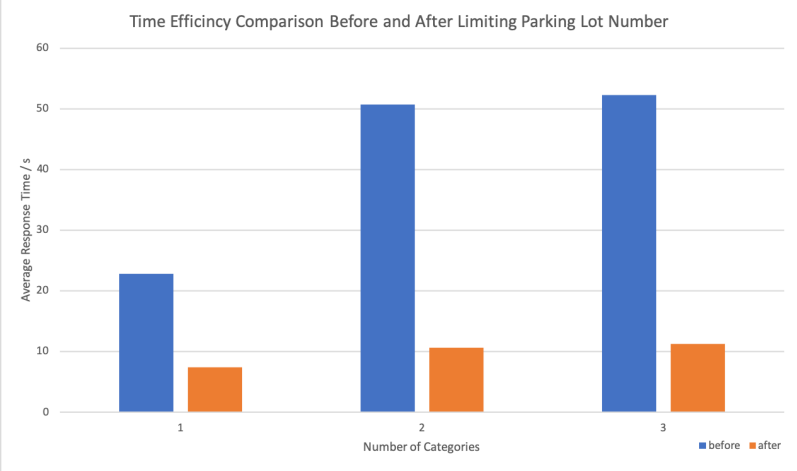

Figure 10. Back End Time Efficiency Improvement

7 CONCLUSION & DISCUSSION

In this project, we designed and implemented a web-based application that design a one-day travel plan for user in Los Angeles. We take from users the type of attractions they want to visit, their departure time and preferred restaurant style in multiple steps. We provide interactive web-interface and display the results of each stage to users. Users can see the current route planning progress clearly and go back to the previous steps and change input for a better travel plan.

Besides flexible and user-friendly interactive interface, we also provide accurate travel planning result. The route is generated based on comprehensive tourist spot data, parking lot data and restaurant data from multiple data sources. Our algorithm outputs reasonable travel plan efficiently in most cases. Our result clearly shows that travel route planning can be made easier by combining parking lot and tourist spots data. Our application not only finds parking lots for users but also organize all attractions in a way such that users can go to the most number of attractions they want to visit at one time without going back to a visited parking lot only to visit an attraction that was previously missed.

The future work of our project will be to improve our algorithm. One way to improve our algorithm is to increase the amount of data. Besides increasing the number of attractions and parking lots, we can also expand the types of data. For instance, we can take gas station and hotel into consideration. Another way to improve our application is to allow back-end to handle more edge cases. For the time being our algorithm tends to find a route such that parking lots and attractions are spare and distribute evenly in different directions. But in some special cases this is not the best strategy. A better solution would be to dynamically choose hyper-parameters and algorithms to decide on how to plan the route.

8 WORK DISTRIBUTION

Our group separate into three subgroups which are front end group, back end group and database group. The front end group take charge of application interface. The back end group take responsible for route algorithm application. And the database group mainly focus on route data and user information storage. All team members have contributed a similar amount of effort.

Front-end	Back-end	Database
Yu-Lin Chung	Ting-Yu Kang	Chuyun Sun
Qian Yu	Xiaofu Niu	Youyi Shi

REFERENCES

[1] Delling, D., Goldberg, A. V., Pajor, T., & Werneck, R. F. (2017). *Customizable Route Planning in Road Networks. Transportation Science*, 51(2), 566–591. doi: 10.1287/trsc.2014.0579

[2] Sanaullah, I., Quddus, M., & Enoch, M. (2016). *Developing travel time estimation methods using sparse GPS data. Journal of Intelligent Transportation Systems*, 20(6), 532–544. doi: 10.1080/15472450.2016.1154764

[3] Han, S.-H., Lee, J.-D., & Ahn, H.-B. (2012). *Geospatial data Acquisition Using the Google Map API. International Journal of Contents*, 8(1), 55–60. doi: 10.5392/ijoc.2012.8.1.055

[4] Tang, K., Chen, S., & Liu, Z. (2018). *Citywide Spatial-Temporal Travel Time Estimation Using Big and Sparse Trajectories. IEEE Transactions on Intelligent Transportation Systems*, 19(12), 4023–4034. doi: 10.1109/tits.2018.2803085

[5] Liebig, T., Piatkowski, N., Bockermann, C., & Morik, K. (2017). *Dynamic route planning with real-time traffic predictions. Information Systems*, 64, 258–265. doi: 10.1016/j.is.2016.01.007

[6] Liebig, T., Piatkowski, N., Bockermann, C., & Morik, K. (2014, March). *Predictive Trip Planning-Smart Routing in Smart Cities*. In EDBT/ICDT Workshops (pp. 331-338).

[7] Letchner, J., Krumm, J., & Horvitz, E. (2006, July). *Trip router with individualized preferences (trip): Incorporating personalization into route planning.* In AAAI (pp. 1795-1800).

[8] David A Fennell (2011). *A tourist space-time budget in the Shetland Islands*

[9] Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., . . . Werneck, R. (2015). *Route Planning in Transportation Networks.*

[10] Chen, C., Zhang, D., Guo, B., Ma, X., Pan, G., & Wu, Z. (2015). *TripPlanner: Personalized Trip Planning Leveraging Heterogeneous Crowdsourced Digital Footprints. IEEE Transactions on Intelligent Transportation Systems*, 16(3), 1259–1273. doi: 10.1109/tits.2014.2357835

[11] Yuan, H., Xu, H., Qian, Y., & Li, Y. (2016). *Make your travel smarter: Summarizing urban tourism information from massive blog data. International Journal of Information Management*, 36(6), 1306–1319. doi: 10.1016/j.ijinfomgt.2016.02.009

[12] Bob McKercher, Noram Shoval, Erica Ng & Amit Birenboim(2011). *First and repeat visitor behaviour: GPS tracking and GIS analysis in Hong Kong*

[13] Deborah Edwards & Tony Griffin (2013). *Understanding tourists' spatial behaviour: GPS tracking as an aid to sustainable destination management*

[14] Modsching, M., Kramer, R., Gretzel, U., & Hagen, K. T. (2006). *Capturing the Beaten Paths: A Novel Method for Analysing Tourists' Spatial Behaviour at an Urban Destination. Information and Communication Technologies in Tourism 2006,* 75–86. doi: 10.1007/3-211-32710-x_15

[15] Bahuleyan, H., & Vanajakshi, L. D. (2017). *Arterial Path-Level Travel-Time Estimation Using Machine-Learning Techniques. Journal of Computing in Civil Engineering*, 31(3), 04016070. doi: 10.1061/(asce)cp.1943-5487.0000644

[16] Kaplan, S., Nielsen, T. A. S., & Prato, C. G. (2016). *Walking, cycling and the urban form: A Heckman selection model of active travel mode and distance by young adolescents. Transportation Research Part D: Transport and Environment*, 44, 55–65. doi: 10.1016/j.trd.2016.02.011

[17] Wang, J., & Cao, X. (2017). *Exploring built environment correlates of walking distance of transit egress in the Twin Cities. Journal of Transport Geography*, 64, 132–138. doi: 10.1016/j.jtrangeo.2017.08.013

[18] Király, A., & Abonyi, J. (2015). *Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google Maps API. Engineering Applications of Artificial Intelligence*, 38, 122–130. doi: 10.1016/j.engappai.2014.10.015

[19] Ruthkoski, T. (2013). *Google Visualization API essentials make sense of your data : Make it visual with the Google Visualization API (Community experience distilled)*. Birmingham, U.K.: Packt Pub.

[20] `https://www.yelp.com/developers`

[21] `https://developers.google.com/`

[22] `https://www.here.com/`

[23] `https://www.mongodb.com/cloud/atlas`