Parking Oriented Travel Route Planner

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

Introduction

In this project, we designed and implemented a full-stack web application project that provides one-day travel route planning for users.

1. The first travel planning application combines attraction finding, route planning, and parking lots searching.
2. 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.

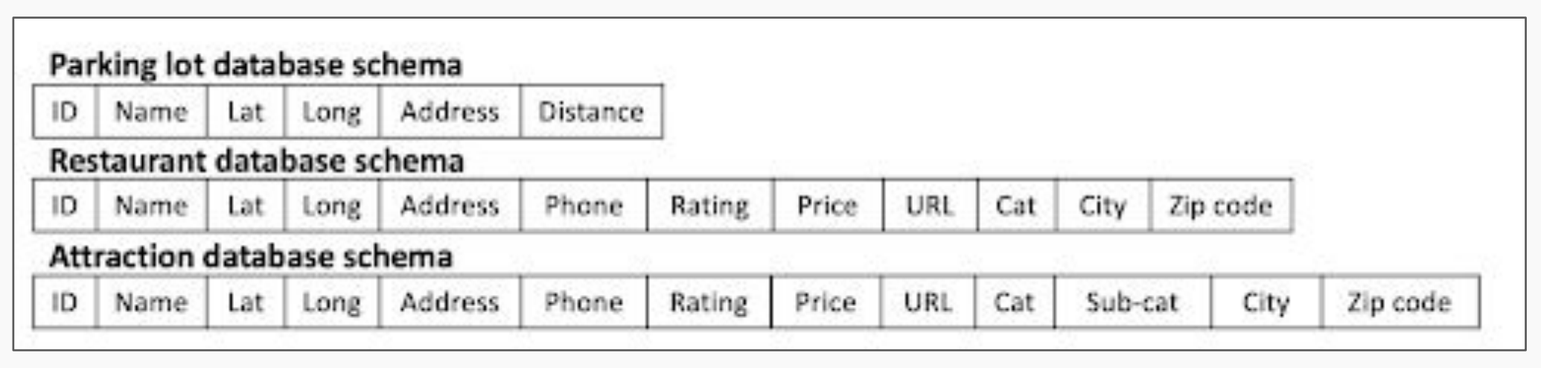
Motivation

1. There are so many tourism tour planning applications in the market, but none of them includes parking information.
2. As more and more people choose self-drive tours, it will be a high demand for an application like this.

Data

The data are combined from multiple platforms. The latitude is restricted between 34.3 and 33.7 and longitude is restricted between -118.15 and -118.67 for Los Angeles area.

Around 2,300 attractions and their locations are obtained from Yelp and Google APIs. Those attractions with are then split into 4 categories and 18 subcategories. Yelp provides information on location, price and rating of restaurants. Total of 4,400 restaurants are split into 4 types of cuisine. Data of parking lot locations and prices are obtained from Here, a parking mobility platform. Around 24,000 parking lots in Los Angeles are used in our app.



Parking lot database schema

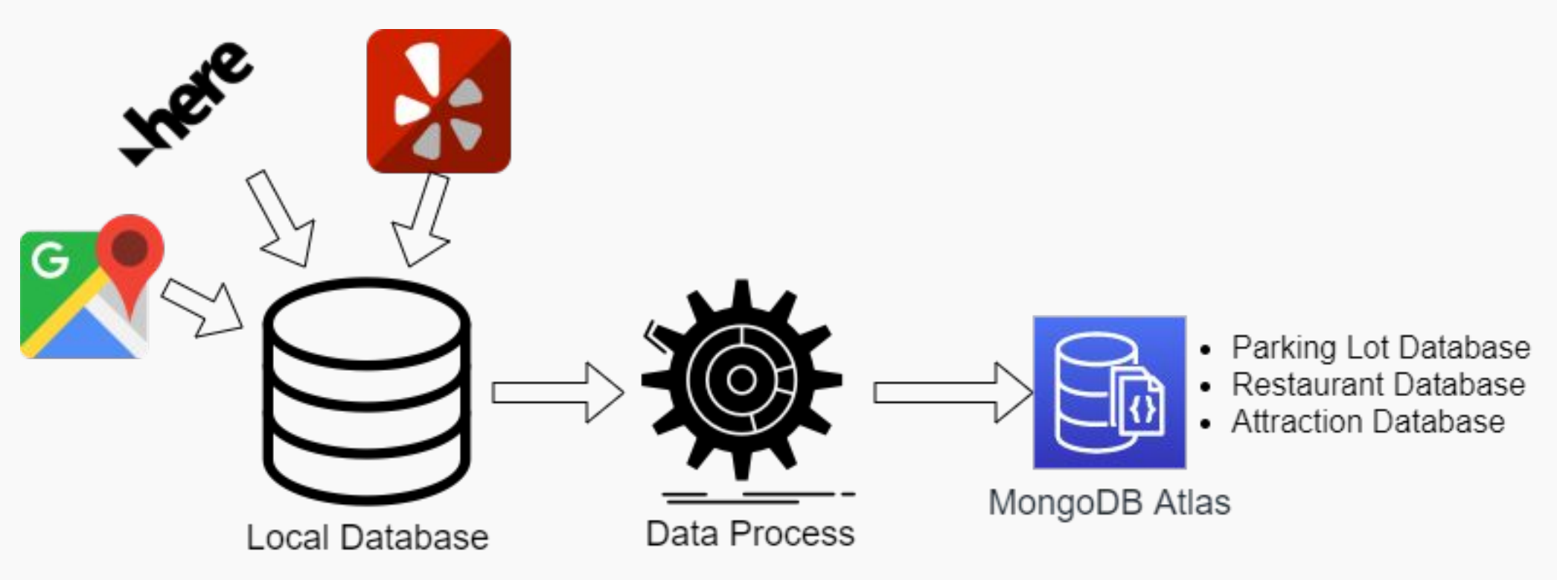
ID	Name	Lat	Long	Address	Distance

Restaurant database schema

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

Attraction database schema

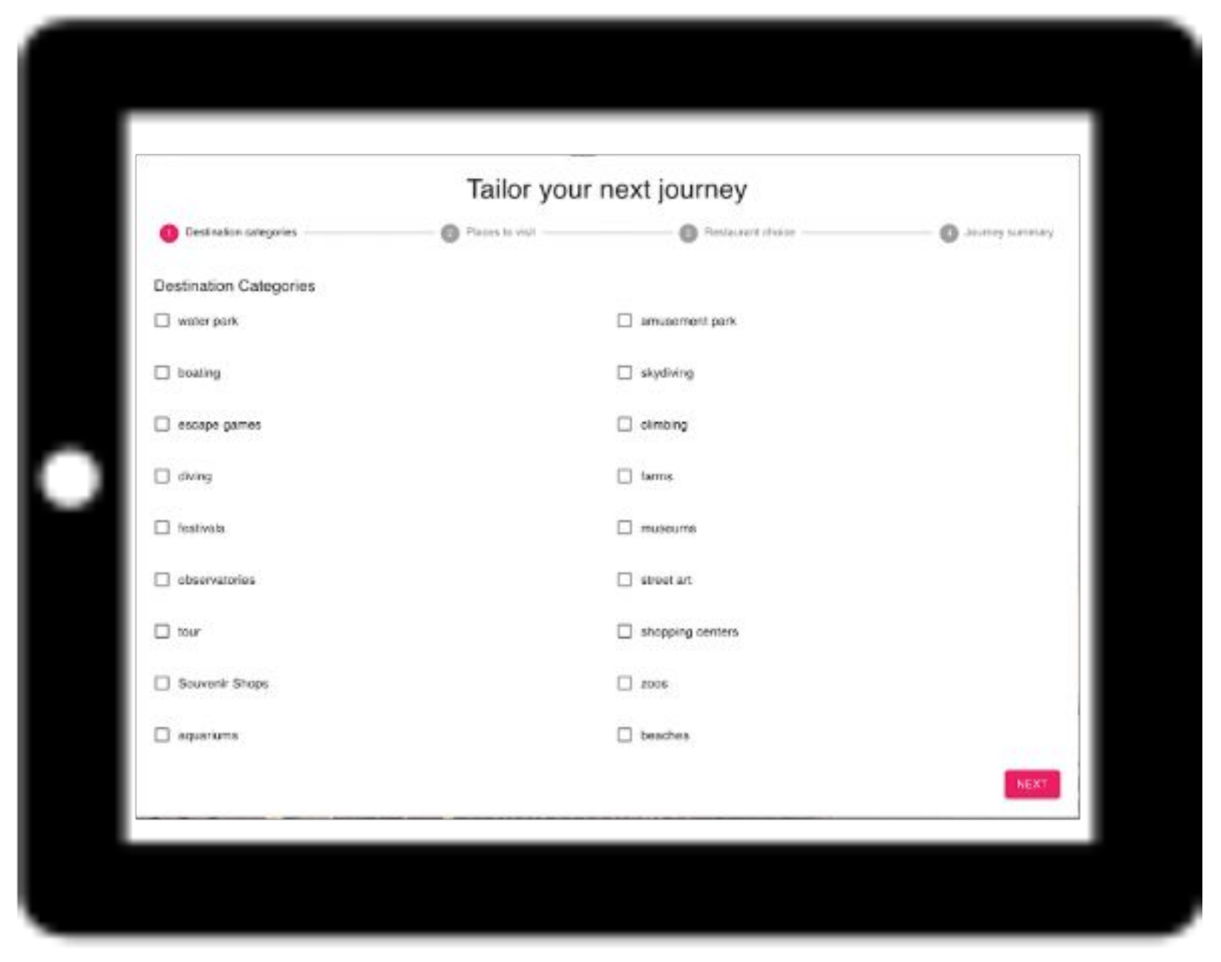
ID	Name	Lat	Long	Address	Phone	Rating	Price	URL	Cat	Sub-cat	City	Zip code

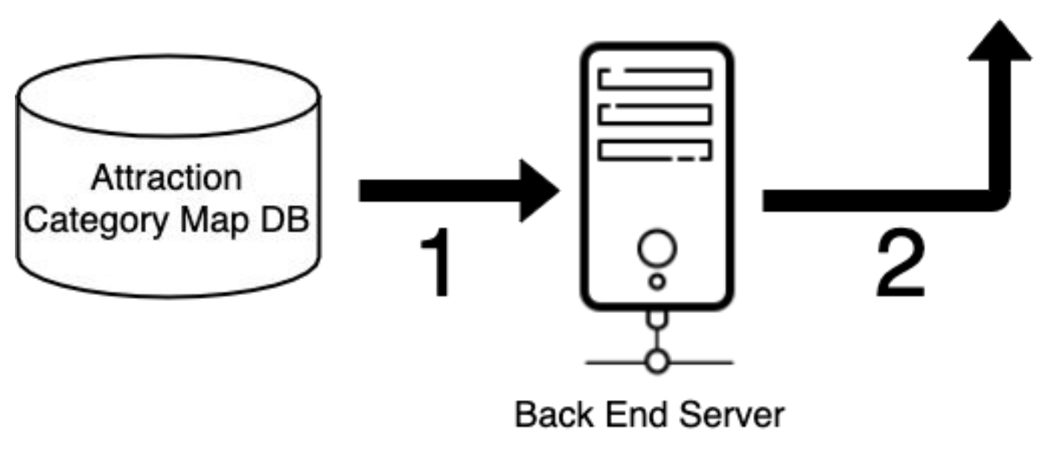


The last step is to setup database by importing our data to MongoDB Atlas cloud server. Three collections are created for data of attractions, restaurants, and parking lots. For attractions and restaurants we also have a collection to map category name to category ID.

Step 1. Preprocess

User checks the attraction types on this page which they are interested in.

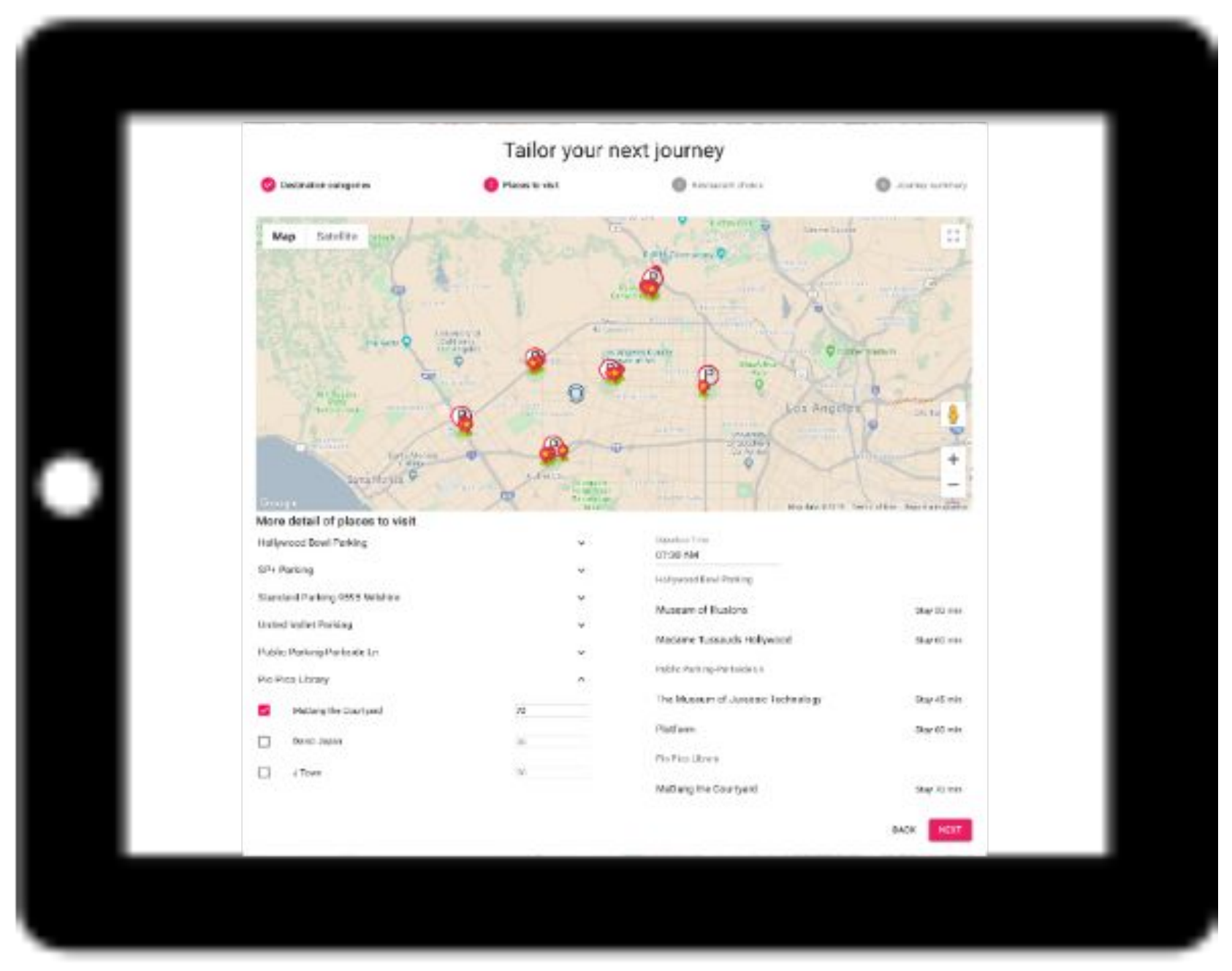


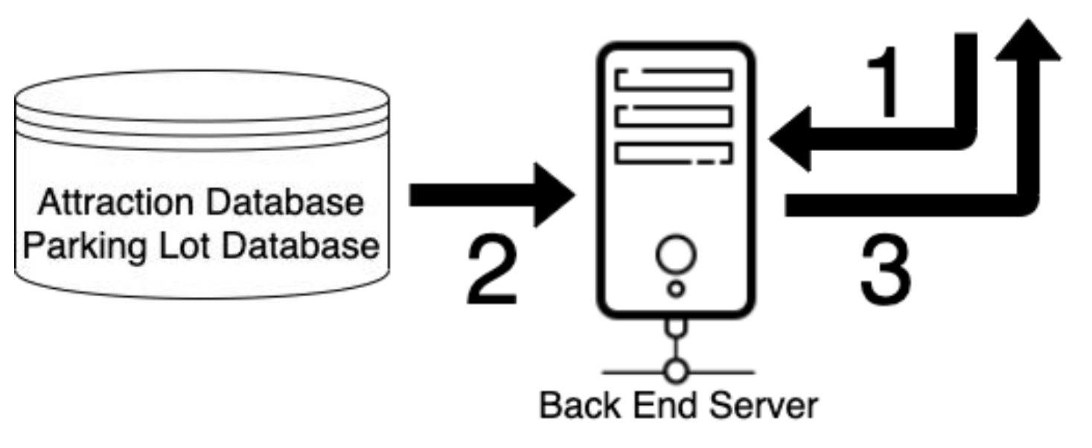


Back-end collects attraction categories from database and sends it to front-end.

Step 2. Attraction Preference

Front-end displays a list of attraction groups and lets user choose attractions they want to go.

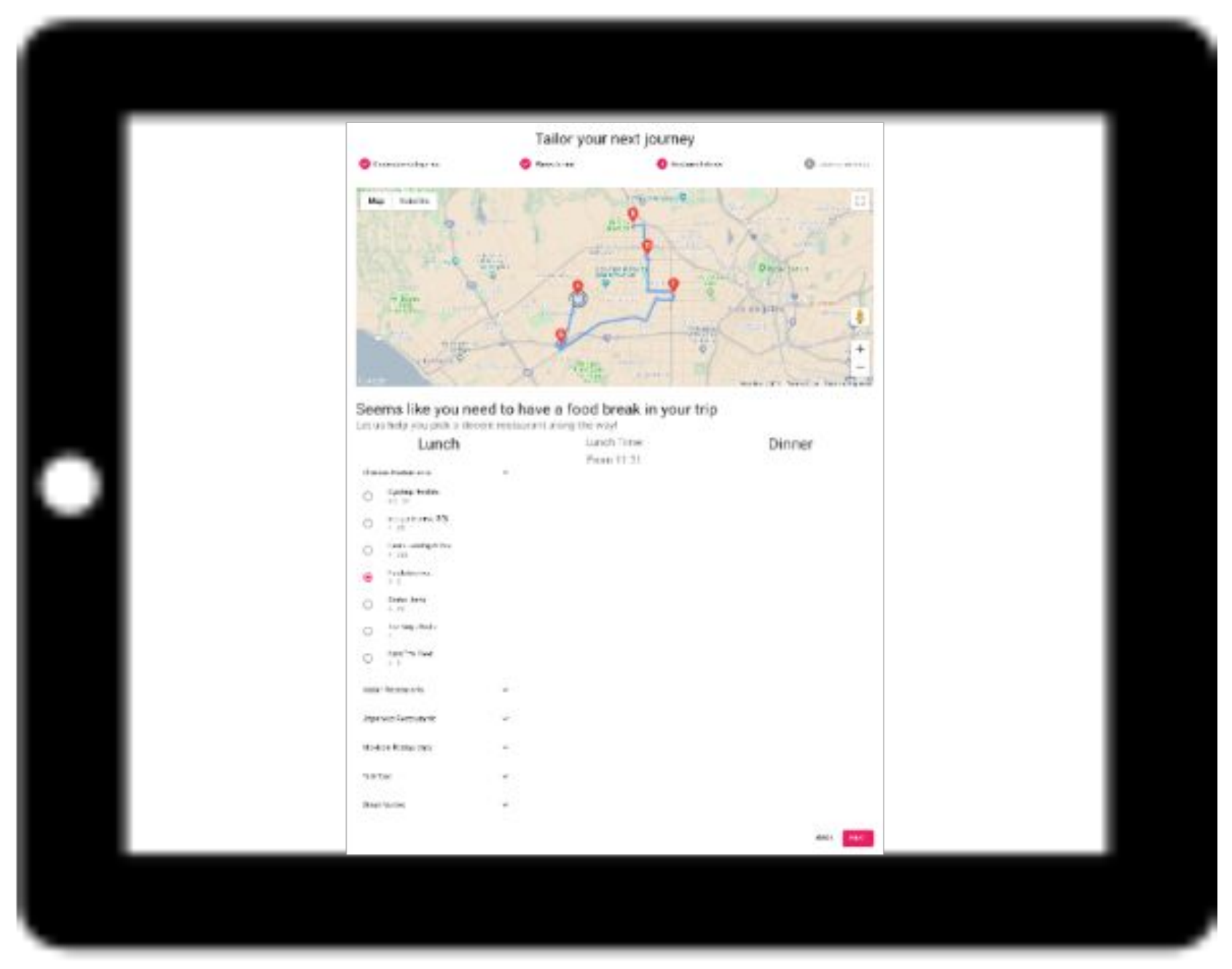


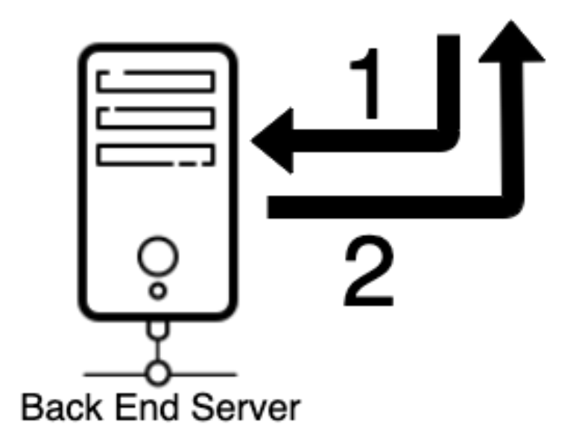


Back-end retrieves data from database and find at the best attraction groups based on user preference.

Step 3. Route planning

Front-end displays the route, enabling user to choose the restaurant type.

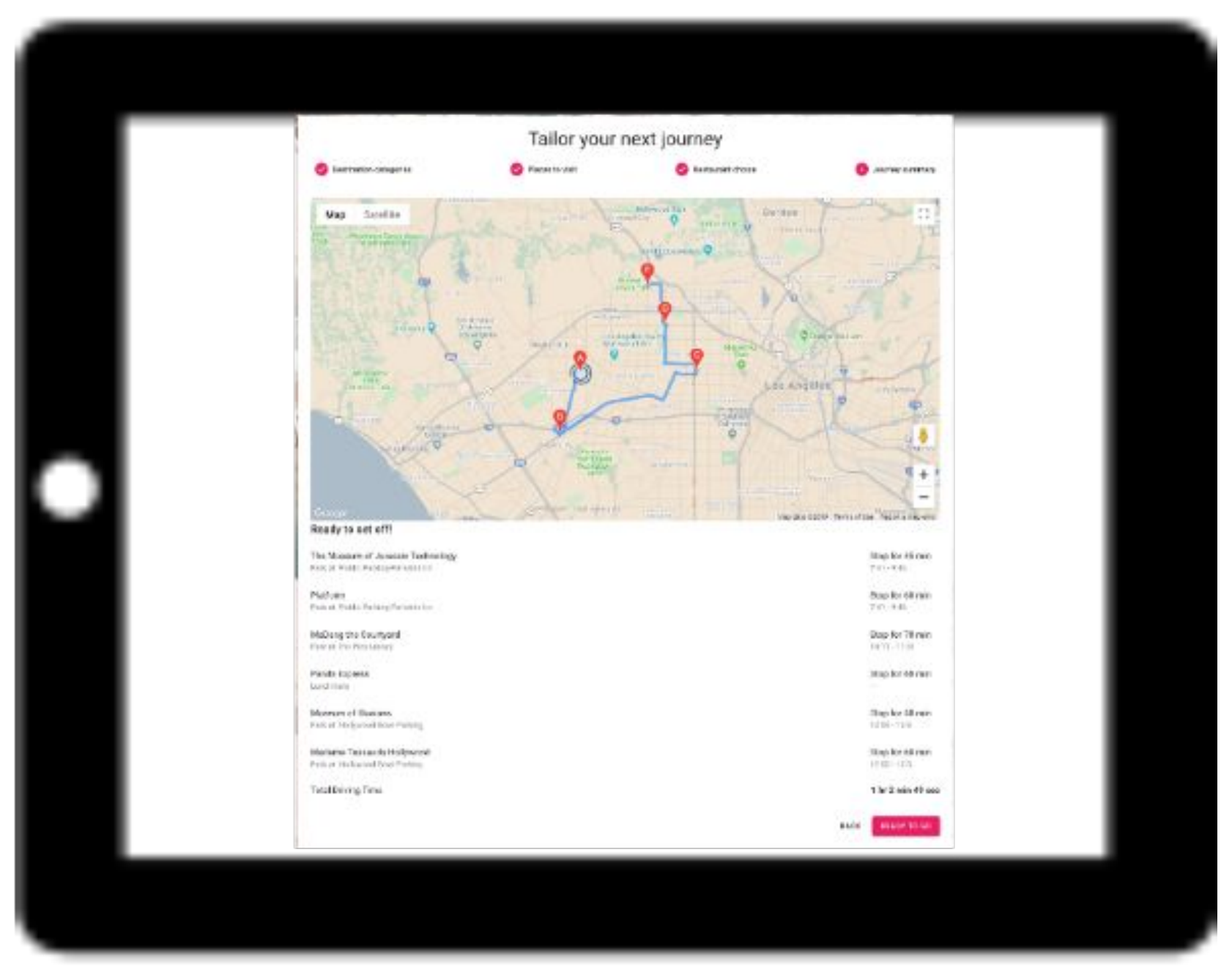


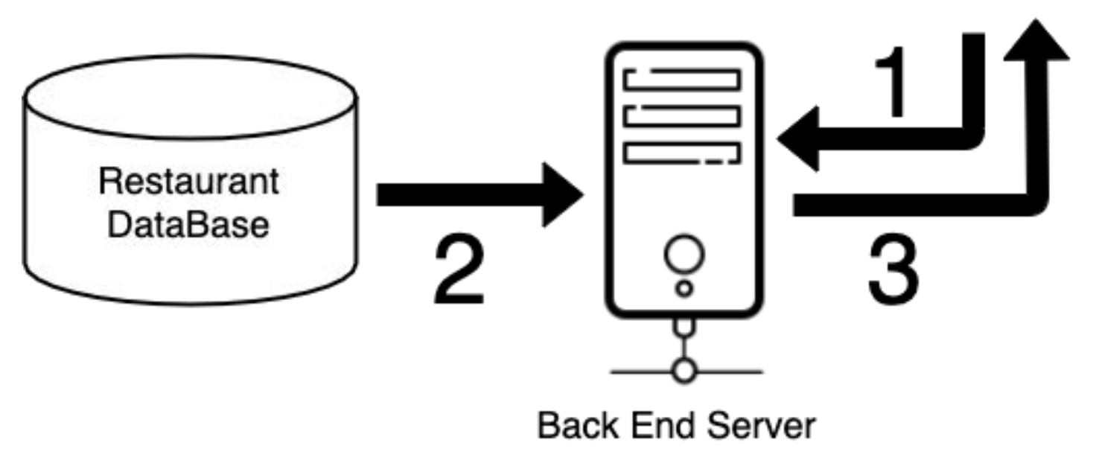


Use DFS to find all possible routes and choose the one with the shortest traveling distance. Find restaurants based on travel time.

Step 4. Restaurant Choices

Front-end displays the finalized route with restaurant chosen by user





Experiment

Database

1. Check data distribution.
2. Parking and attraction location

Front-end

1. Interface construction check experiment
2. Interface route check experiment

Back-end

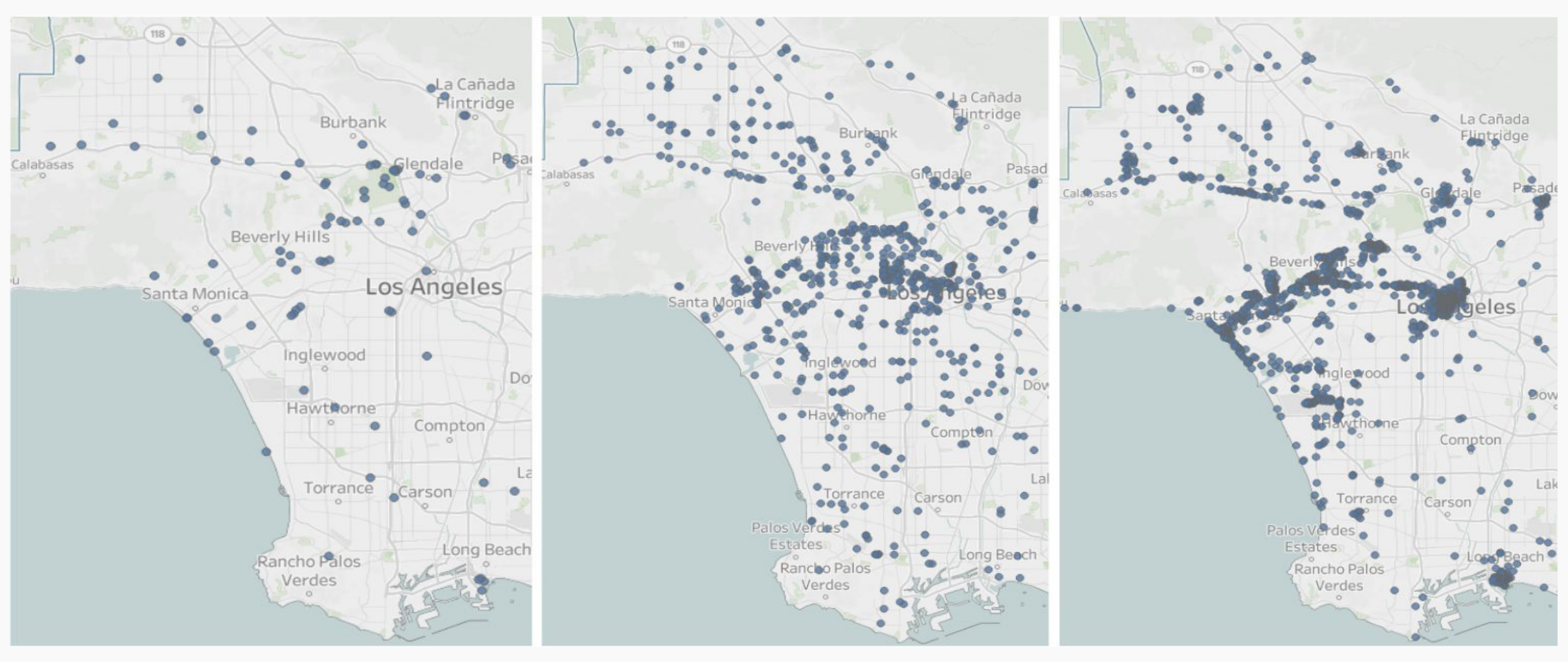
1. Euclidean Algorithm test accuracy
2. Euclidean Algorithm time efficiency

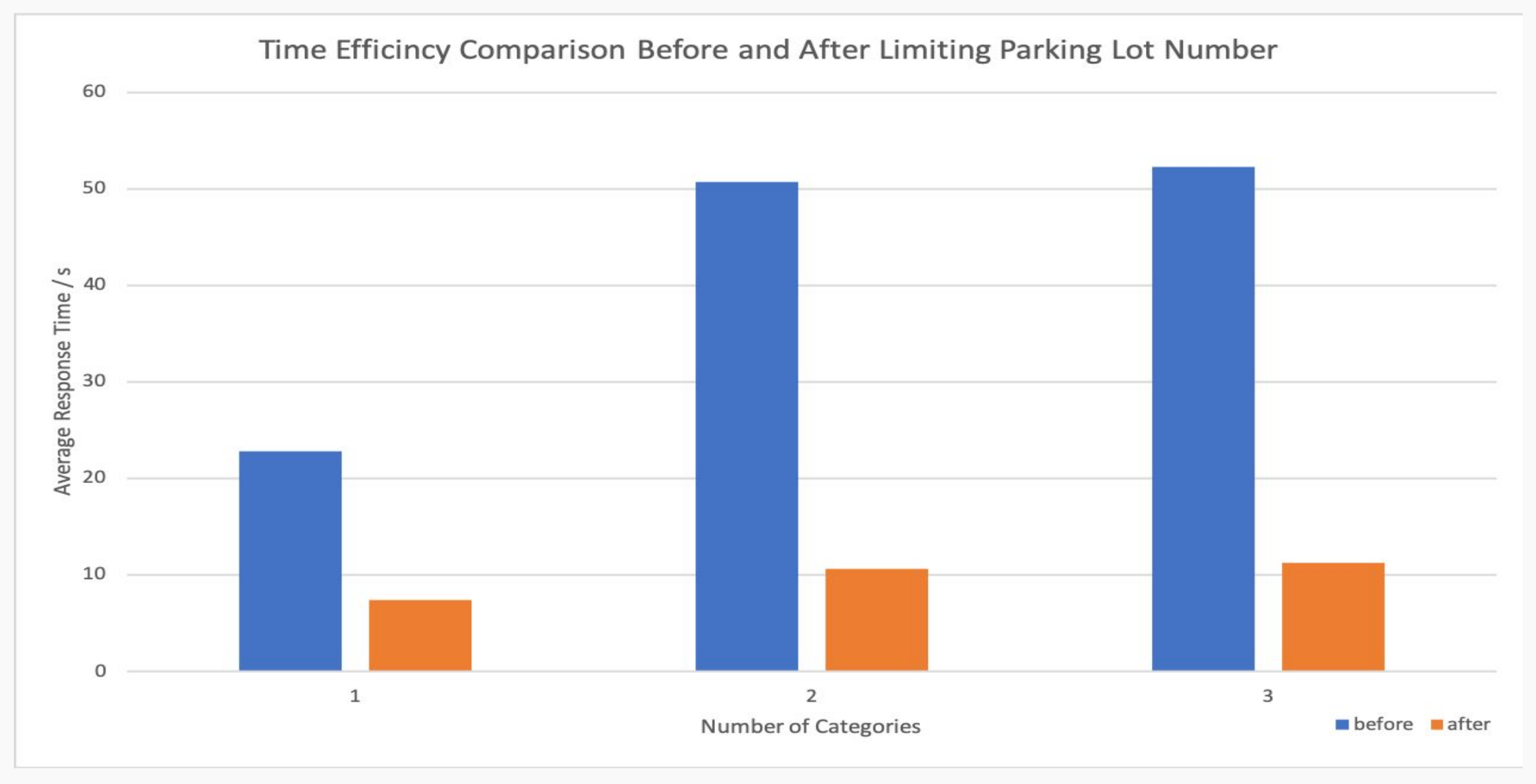
Group9

Result & Evaluation

Database

We plot our attractions, restaurants, and parking lots to observe the density of data points.





Front-end

Handle different window sizes and different number of data provides on our web page.

Our route display section give tester a direct and reliable route recommendation.

Back-end

1. The number of attraction groups varies according to current location and preferred attraction types.
2. Next we evaluated the efficiency of algorithm in terms of response time. The result shows that reducing number of parking lots successfully increase the time efficiency.

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