Final Project Report

Kuan-Hui Lin (5528766160) Peiying Lyu (8109407016)

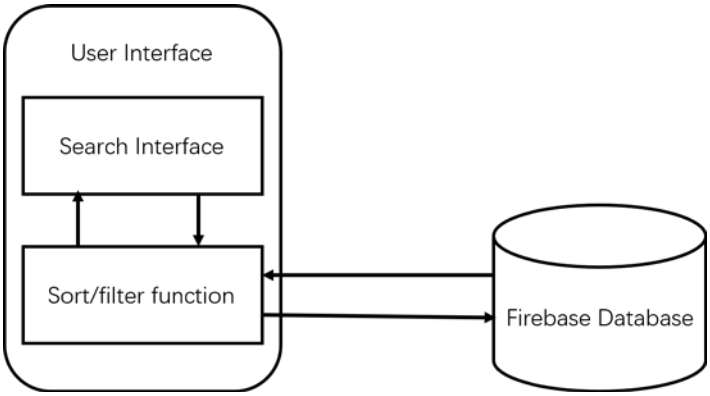
1. Project Idea

Our work is going to provide data information for Uber so that Uber can consolidate its customers base. Through understanding customers' use, Uber can offer specific preferential schemes to special customers groups, such as business, so that these customers can enjoy their preferential policies and continue supporting the use of Uber.

2. UI Architecture

We design user interface by front-end development tools like HTML, CSS, JavaScript and the common framework, **Bootstrap**. On the top of screen, we put our name of topic, My Uber, and then we design some function buttons, like search buttons and filter buttons, which can let users do specific search, and we also design sorting up and down icons by using free icons from Font Awesome. In addition, we also add data visualization by using **ECharts**, which provide an accessible way to see patterns in data. Besides, we also use **Firebase** to be our database.

A. Architecture Diagram



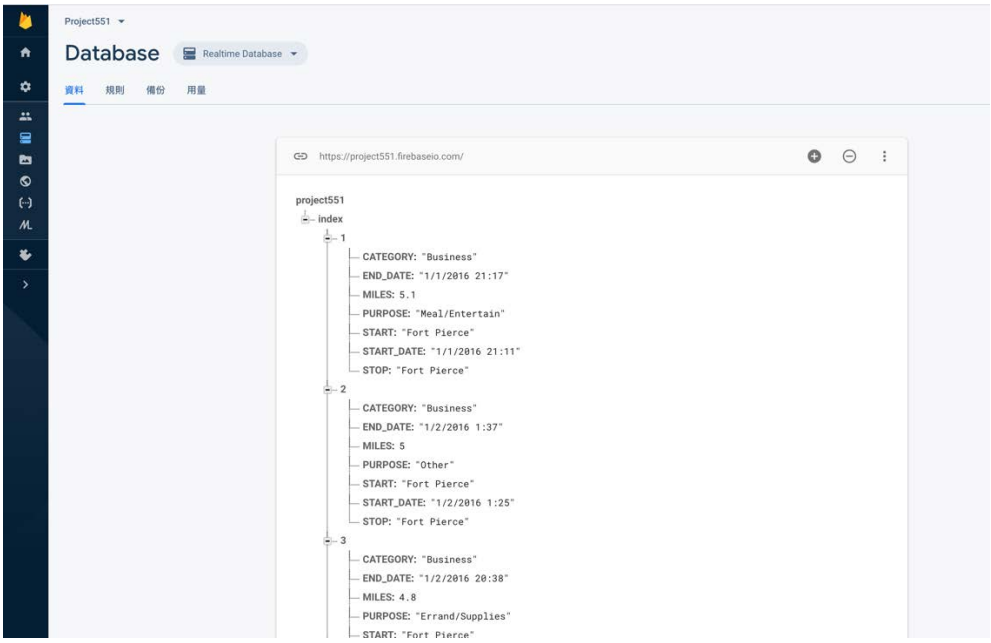
I. Implementation: Firebase

First, we upload our dataset to Firebase by using Python module, request, and then the dataset will be represented in JSON format. Because there is no unique value of each attribute in our dataset, we give each instance a unique ID number like 1, 2, 3…etc. to be a key in Firebase.

a. Display Python requests package:

```python
from collections import defaultdict
import urllib.parse
import pandas as pd
import csv, json
import sys
import requests
import re

#enter the file name and key name in command line
filename = sys.argv[1]
csvFilePath = filename
data = pd.read_csv(csvFilePath)
df = pd.DataFrame(data)
d = df.set_index('User_ID').to_dict(orient='index')

jsondata = json.dumps(d)

url = 'https://project551.firebaseio.com/index.json'
response = requests.put(url,jsondata)
print(json.dumps(response.json(), indent=2, separators=(',\t',':')))
```

b. Display our database – Firebase:

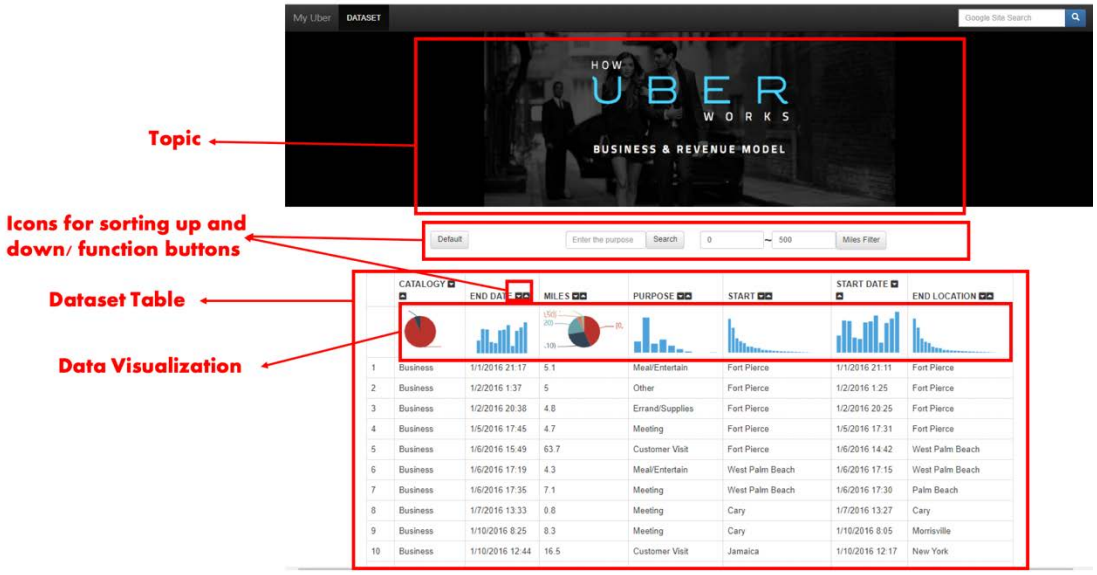


II. Front-end development tools/framework: HTML, CSS, JavaScript, Bootstrap

In order to design a user-friendly interface, we utilize a free and open-source CSS framework, Bootstrap, which develop with HTML, CSS, and JS. First, we design a navigation bar that is placed at the top of the page, and also add a Google search box on the top right-hand side, which can let users click the search engine to search what they want, and then we design menu on the top of the page. Second, in the middle of the page, we put our topic name which can clearly let users know what this website is doing. In addition, we design the interface

including search and filter buttons. About data visualization, this part will be illustrated later. Finally, we place **Font Awesome** icons on top of each column of the table to be sorting up and down click icons.

a. Our website screenshot:



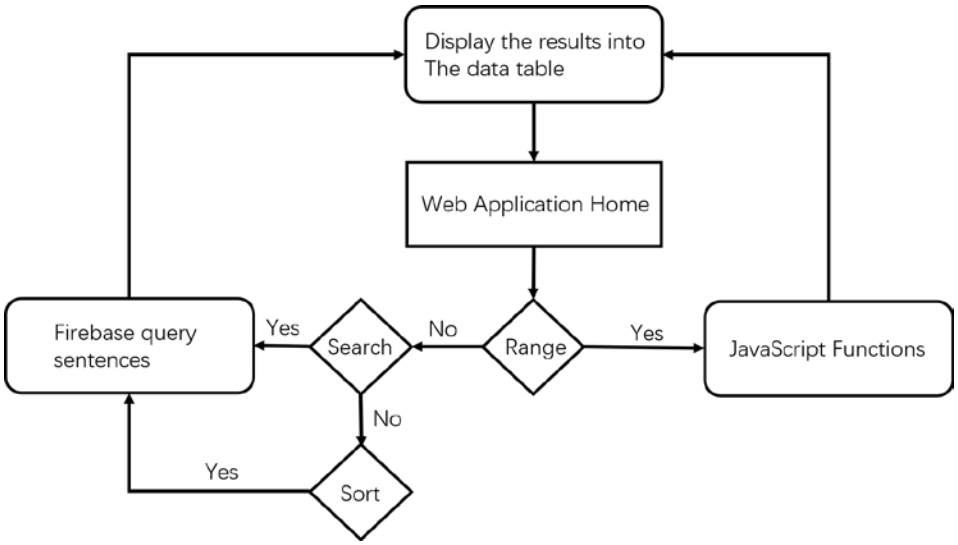
b. Display how to implement the interface and function buttons

```
<div id="menu1" class="tab-pane fade in active">
    <div style="text-align: center;background: black;">
        <img src="uber_pic.png">
    </div>
    <div class="row" style="margin-top: 3%;margin-left: 1%;">
        <div class="col-xs-2" style="display: inline-flex; margin-left:16%;">
            <button type="button" class="btn btn-default" onclick="default_()">Default</button>
        </div>
        <div class="col-xs-2" style="display: inline-flex;">
            <input id= "search_item_2" type="search" class="form-control" placeholder="Enter the purpose" name="qfront">
            <button class="btn btn-default" type="submit" onclick="search_purpose()">Search</button>
        </div>
        <div class="col-xs-3" style="display: inline-flex;">
            <input id="small_" type="number" class="form-control" placeholder="Enter a num " value=0>
            <p style="font-size: 25px;margin: unset;">~</p>
            <input id="big_" type="number" class="form-control" placeholder="Enther a num" value=500>
            <button class="btn btn-default" type="submit" onclick="range_()">Miles Filter</button>
        </div>
    </div>
```

c. Display how to add sort up and down icons and how to give these icons function.

```
<tr>
    <th> </th>
    <!— <th>sequence <i class="fas fa-caret-square-down" onclick=""></i><i class="fas fa-caret-square-up" onclick=""></i></th> —>
    <th>CATALOGY <i class="fas fa-caret-square-down" onclick="catalog_sort_down()"></i><i class="fas fa-caret-square-up" onclick="catalog_sort_up()"></i></th>
    <th>END DATE <i class="fas fa-caret-square-down" onclick="enddate_sort_down()"></i><i class="fas fa-caret-square-up" onclick="enddate_sort_up()"></i></th>
    <th>MILES <i class="fas fa-caret-square-down" onclick="mile_sort_down()"></i><i class="fas fa-caret-square-up" onclick="mile_sort_up()"></i></th>
    <th>PURPOSE <i class="fas fa-caret-square-down"onclick="purpose_sort_down()"></i><i class="fas fa-caret-square-up" onclick="purpose_sort_up()"></i></th>
    <th>START <i class="fas fa-caret-square-down" onclick="start_sort_down()"></i><i class="fas fa-caret-square-up" onclick="start_sort_up()"></i></th>
    <th>START DATE <i class="fas fa-caret-square-down" onclick="startdate_sort_down()"></i><i class="fas fa-caret-square-up" onclick="startdate_sort_up()"></i></th>
    <th>END LOCATION <i class="fas fa-caret-square-down" onclick="destination_sort_down()"></i><i class="fas fa-caret-square-up" onclick="destination_sort_up()"></i>
    </th>
</tr>
```

B. Design Flow Diagram



3. Functions Implementation

A. Basic Structures

a. Displaying firebase data to html data table

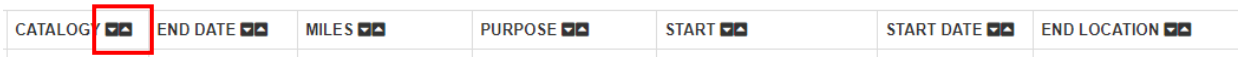
```
524 var database = firebase.database();
525 var myuber = database.ref('/index/');
526 var djson;
527
528 myuber.on('value', function(snapshot){
529     djson = JSON.parse(JSON.stringify(snapshot.val()));
530     console.log(djson);
531     var tr = "";
532     for(x in djson){
533         if(djson[x] != null){
534             tr += "<tr>";
535             tr += "<td>"+x+"</td>";
536             tr += "<td>"+djson[x]["CATEGORY"]+"</td>";
537             tr += "<td>"+djson[x]["END_DATE"]+"</td>";
538             tr += "<td>"+djson[x]["MILES"]+"</td>";
539             tr += "<td>"+djson[x]["PURPOSE"]+"</td>";
540             tr += "<td>"+djson[x]["START"]+"</td>";
541             tr += "<td>"+djson[x]["START_DATE"]+"</td>";
542             tr += "<td>"+djson[x]["STOP"]+"</td>";
543             tr += "</tr>";
544         }
545     }
546     document.querySelector("table tbody").innerHTML = tr;
547 });
```

Line 524 – 529: Firebase Filtering data

Line 533 – 543: First, traversing each line; then, determine if the instance is NULL; finally, according to attribute names, putting each string into each column of each row.

B. Sorting Functions

Users can implement the function by using buttons near columns names.



a. Numerical

Attribute "**MILES**" has been chosen, which is the only one numerical attribute in the dataset. First, calling the whole date set to local HTML; then, before putting instances into a data table, we use JavaScript sorting functions to handle them.

```
// #ascending
function up_compare(property){
    return function(obj1, obj2) {
        var value1 = obj1[property];
        var value2 = obj2[property];
        return value1 - value2;
    }
}
// #descending
function down_compare(property){
    return function(obj1, obj2) {
        var value1 = obj1[property];
        var value2 = obj2[property];
        return value2 - value1;
    }
}
```

b. Non-numerical

Attributes "**CATEGORY**," **END DATE**," "**PURPOSE**," **START LOCATION**," "**START DATE**," **STOP LOCATION**" have been chosen. As the above part, we use JavaScript sorting functions to handle them but different sorting functions.

```
function up_string(property){
    return function(obj1, obj2) {
        var value1 = obj1[property];
        var value2 = obj2[property];
        return (value1 > value2)?1:-1;
    }
}
function down_string(property){
    return function(obj1, obj2) {
        var value1 = obj1[property];
        var value2 = obj2[property];
        return (value2 > value1)?1:-1;
    }
}
```

C. Filter Functions

a. Numerical

Users can implement this function by using the following buttons.

The default minimum and maximum have set up while Implementing this function, **0** and **500**. When users input only one number, the other one will call the default

number.

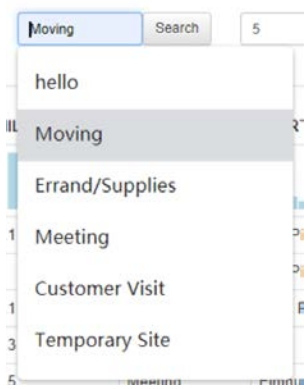
```
var small_ = 0;
var big_ = 500;
if(document.getElementById("small_").value != null ){
    small_ = document.getElementById("small_").value;}
console.log(small_)

if(document.getElementById("big_").value != null ){
    big_ = document.getElementById("big_").value;}
```

After making sure there is a range, there is a **condition function** to pick up the valid instances to display on the data table.

```
if(djson[x]["MILES"] >= small_ && djson[x]["MILES"] <= big_){
```

b. Non-numerical

Attribute "**PURPOSE**" has been used. Users can input the keyword from the keyboard, and the following recommend menu.



After clicking the button, **the search Item** will send to the firebase by using firebase query sentences.

```
var srearchItem = document.getElementById("search_item_2").value;
console.log(srearchItem)
var purpose_ = database.ref('/index').orderByChild('PURPOSE').equalTo(srearchItem);
```

Then, if the **search Item** is valid, firebase can return matched instances; otherwise, it shows nothing.

1	Personal	7/18/2016 10:49	4.1	Moving	Cary	7/18/2016 10:37	Morrisville
2	Personal	7/18/2016 11:15	6.1	Moving	Morrisville	7/18/2016 10:54	Cary
3	Personal	7/18/2016 11:36	3.3	Moving	Northwoods	7/18/2016 11:25	Preston
4	Personal	7/18/2016 11:56	4.7	Moving	Preston	7/18/2016 11:40	Whitebridge

D. Data Visualization

In this part, Pie charts and Bar charts.

We used **ECharts** to realize data visualization. Before drawing, a **DOM** container for ECharts has been set up. Then, initializing an instance of echarts with **echarts.init** method and generate a bar chart and pie chart with the **setOption** method.

```
<div id="main" style="width: 80px;height:80px;"></div>
<script type="text/javascript">
    var myChart = echarts.init(document.getElementById('main'));
    var option = {
        title: {},
        tooltip: {},
        legend: {
            show: false
        },
        xAxis: {
            show: false
            // data: ["Business","Personal"]
        },
        yAxis: {
                show: false
            },
        series: [{
            type: 'pie',
            data: [{value:1078, name:'Business'},
                    {value:77, name:'Personal'}]
            // color:['Lightblue']
        }]
    };
    myChart.setOption(option);
```

E. Default

Considering that users may be confused after too many operations, we implemented a default button to return to the whole full data set.

4. Responsibility

Application Interface	Kuan-Hui Lin
Database management	Kuan-Hui Lin
Data visualization	Peiying Lyu
Functions implement	Peiying Lyu