Assignment #3 Documentation

Yuqi Zhou

23 March

server.js

1) Add three post functions to query data from the database

```
router.route('/stations/selected').post((req, res) => {
    var str = JSON.stringify(req.body, null, 4);
    for (var i = 0,len = stations_found.length; i < len; i++) {

        if ( stations_found[i].id === req.body.id ) { // strict ...
            equality test

            station_selected = stations_found[i];

            break;
        }
    }

    const query1 = {
        // give the query a unique name
        name: 'fetch-divvy-selected-station',
        text: ' SELECT * FROM divvy_stations_logs WHERE id = $1 ...
            AND lastcommunicationtime >= (NOW() - INTERVAL \'1 ...
            hours\' ) order by lastcommunicationtime',
        values: [station_selected.id]
    }

    find_selected_stations_from_divvy(query1).then(function ...
        (response) {
        var hits = response;
        res.json({'station_selected_info': 'Added successfully'});
    });

});
```

Order data by the last communication time

app.modue.ts

1) Add LineChartDivvyComponent component into app.module.ts

```
import { LineChartDivvyComponent } from ...
    './components/line-chart-divvy/line-chart-divvy.component';
```

2) Add path for the new component in routes.

```
const routes: Routes = [
  { path: 'find', component: FindComponent},
  { path: 'list_of_places', component: ListOfPlacesComponent},
  { path: 'list_of_stations', component: ListOfStationsComponent},
  { path: 'line_chart_divvy', component: LineChartDivvyComponent},

  { path: '', redirectTo: 'find', pathMatch: 'full'}
];
```

list-of-stations

1). Add Line Chart buttons behind every record of stations.

```
<ng-container matColumnDef="lineChart">
        <th mat-header-cell *matHeaderCellDef ...
            class="mat-column-right"></th>
        <td mat-cell *matCellDef="let element" ...
            class="mat-column-right">
          <button mat-raised-button color="primary" ...
              (click)="findSelectedStations(element.id)">Line ...
              Chart</button>
        </td>
      </ng-container>
```

2). Add Back button. Go back to /list_of_places page.

```
<button mat-raised-button color="primary" ...
    routerLink="/list_of_places">
        Back
             
        <mat-icon>search</mat-icon>
      </button>
```

places.services.ts

1). Get one hour data

```
findSelectedStations(id) {
    const find_selected_stations = {
      id: id
    };

    //console.log("find selected stations",id)
    var str = JSON.stringify(find_selected_stations, null, 2);

    return this.http.post(`${this.uri}/stations/selected`, ...
        find_selected_stations, httpOptions);
  }
```

2). Get 24 hours data

```
  findSelectedStationsTwentyFourHour(id) {
    const find_selected_stations = {
      id: id
    };

    //console.log("find selected stations",id)
    var str = JSON.stringify(find_selected_stations, null, 2);

    return ...
        this.http.post(`${this.uri}/stations/selected_twenty_four_hour`, ...
        find_selected_stations, httpOptions);
  }
```

3). Get seven days data

```
findSelectedStationsSevenDay(id) {
    const find_selected_stations = {
      id: id
    };
```

```
5
6       //console.log("find selected stations",id)
7       var str = JSON.stringify(find_selected_stations, null, 2);
8
9       return ...
            this.http.post(`${this.uri}/stations/selected_seven_day`, ...
            find_selected_stations, httpOptions);
10    }
```

line-chart-divvy.component

1). In ngOnInit() function, get data from place service(this.placesService.getSelectedStation()) and then use the data to draw the Line Chart. The past hour data is plotted by default for the user.

```
1     ngOnInit() {
2       this.averageHourNumber = 0;
3       this.averageDayNumber = 0;
4       this.initSvg();
5       this.initAxis();
6       this.placesService
7         .getSelectedStation()
8         .subscribe((data: Station[]) => {
9           this.thisID = data[0].id
10          this.stationData = data
11          //this.sortData()
12          this.stationSelected$ = of(this.stationData)
13          this.drawChart()
14        });
15
16      this.subscribeIntervalOneHour()
17    }
```

2). Subscribe real-time updating for one hour data.

```
1   subscribeIntervalOneHour() {
2       this.timeTitle = 'One Hour'
3       if (this.subscription) {
4         this.subscription.unsubscribe()
5       }
6       this.updateDateTwoMinutes();
7       this.subscription = interval(1000 * 1 * 60).subscribe(() => {
8         console.log("calling 1");
9         this.updateDateTwoMinutes();
10      });
11    }
```

Firstly, unsubscribe.

3). Subscribe real-time updating for twenty four hours data.

```
subscribeIntervalTwentyFourHour() {
    this.timeTitle = 'Twenty Four Hours'
    if (this.subscription) {
      this.subscription.unsubscribe()
    }
    this.updateDateTwoMinutesTwentyFourHour();
    this.subscription = interval(1000 * 1 * 60).subscribe(() => {
      console.log("calling 24")
      this.updateDateTwoMinutesTwentyFourHour();
    });
  }
```

4). Subscribe real-time updating for seven days data.

```
  subscribeIntervalSevenDay() {
    this.timeTitle = 'Seven Days'
    if (this.subscription) {
      this.subscription.unsubscribe()
    }
    this.updateDateTwoMinutesSevenDay();
    this.subscription = interval(1000 * 1 * 60).subscribe(() => {
      console.log("calling 7");
      this.updateDateTwoMinutesSevenDay();
    });
  }
```

5). Subscribe real-time updating for SMA data

```
subscribeIntervalSMA() {
    this.timeTitle = 'SMA CHART'
    if (this.subscription) {
      this.subscription.unsubscribe()
    }
    this.updateDateSMA();

    this.subscription = interval(1000 * 1 * 60).subscribe(() => {
      console.log("calling SMA");
      this.updateDateSMA();
    });
  }
```

6). Update one hour data and the line chart.

```
updateDateTwoMinutes() {
    this.placesService
      .findSelectedStations(this.thisID)
      .subscribe(() => {
        //console.log("updateDateTwoMinutes:", ...
            this.stationData[1].id)
        this.getSelectedStation()
      });
  }
```

7). Update twenty four hours data and the line chart

```
updateDateTwoMinutes() {
    this.placesService
      .findSelectedStations(this.thisID)
      .subscribe(() => {
        //console.log("updateDateTwoMinutes:", ...
            this.stationData[1].id)
        this.getSelectedStation()
      });
  }
```

8). Update seven days data and the line chart.

```
 updateDateTwoMinutesSevenDay() {
    this.placesService
      .findSelectedStationsSevenDay(this.thisID)
      .subscribe(() => {
        //console.log("updateDateTwoMinutes:", ...
            this.stationData[1].id)
        this.getSelectedStation()
      });
  }
```

9). Update SMA data and the line chart.

```
  updateDateSMA() {
    this.placesService
      .findSelectedStations(this.thisID)
      .subscribe(() => {
        this.placesService
          .getSelectedStation()
          .subscribe((data: Station[]) => {
            this.stationData = data
            //this.sortData()
            this.setMoveAverageHour()
            console.log("hour", this.moveAverage)
            this.stationSelected$ = of(this.stationData)
```

```
13            });
14          this.placesService
15            .findSelectedStationsTwentyFourHour(this.thisID)
16            .subscribe(() => {
17              this.getSelectedStationDay()
18              //console.log("day", this.moveAverageDay)
19            });
20        });
21    }
```

10). Calculate the one hour moving average

```
1   setMoveAverageHour() {
2      this.averageHourNumber = 0;
3      this.moveAverage = [];
4      for (let i = 0; i < this.stationData.length; i++) {
5        let temp: moveAverage = {} as any;
6        this.averageHourNumber = this.averageHourNumber + ...
             this.stationData[i].availableDocks.valueOf();
7        temp.availableDocks = Number(this.averageHourNumber / (i + ...
             1));
8        temp.lastCommunicationTime = ...
             this.stationData[i].lastCommunicationTime;
9        this.moveAverage.push(temp);
10     }
11    }
```

11). Calculate the 24 hours moving average

```
1   setMoveAverageDay() {
2      this.averageDayNumber = 0;
3      this.moveAverageDay = [];
4      for (let i = 0; i < this.stationDataDay.length; i++) {
5        let temp: moveAverage = {} as any;
6        this.averageDayNumber = this.averageDayNumber + ...
             this.stationDataDay[i].availableDocks.valueOf();
7        temp.availableDocks = Number(this.averageDayNumber / (i + 1));
8        temp.lastCommunicationTime = ...
             this.stationDataDay[i].lastCommunicationTime;
9        this.moveAverageDay.push(temp);
10     }
11    }
```

12). Draw the line chart

```
1   drawChart() {
2      this.drawAxis();
3      this.drawLine();
```

```
4    }
```

13). Update the real-time line chart

```
1    updateChart() {
2      var body = d3.select('body').transition();
3      body.selectAll(".d-inline-block")
4        .style("opacity", 1);
5
6      this.x.domain(d3Array.extent(this.stationData, (d) => new ...
           Date(d.lastCommunicationTime.valueOf())));
7      this.y.domain([0, d3Array.max(this.stationData, (d) => ...
           Number(d.availableDocks.valueOf())) + 5]);
8
9      /* this.line.x((d: any) => this.x(new ...
           Date(d.lastCommunicationTime.valueOf())))
10       .y((d: any) => this.y(d.availableDocks.valueOf())) */
11
12     var svg = d3.select('svg').transition();
13     svg.selectAll(".line")
14       .duration(750)
15       .attr("d", this.line(this.stationData))
16
17     svg.selectAll(".line1")
18       .style("opacity", 0)
19
20     svg.selectAll(".line2")
21       .style("opacity", 0)
22
23     svg.select(".axis.axis--x") // change the x axis
24       .duration(750)
25       .call(this.xAxis);
26
27     svg.select(".axis.axis--y") // change the y axis
28       .duration(750)
29       .call(this.yAxis);
30   }
```

14). Update the SMA line chart

```
1  updateChartSMA() {
2      var body = d3.select('body').transition();
3      body.selectAll(".d-inline-block")
4        .style("opacity", 0);
5
6      /* this.line.x((d: any) => this.x(new ...
           Date(d.lastCommunicationTime.valueOf())))
7        .y((d: any) => this.y(d.availableDocks.valueOf())) */
8      this.cutMoveAverageDay = ...
           this.moveAverageDay.slice(-this.stationData.length)
9      //console.log(this.cutMoveAverageDay)
```

```
10
11      this.x.domain(d3Array.extent(this.stationData, (d) => new ...
            Date(d.lastCommunicationTime.valueOf())));
12      this.y.domain([0, d3Array.max(this.cutMoveAverageDay, (d) => ...
            Number(d.availableDocks.valueOf())) + 5]);
13
14      var svg = d3.select('svg').transition();
15
16      svg.selectAll(".line")
17        .duration(750)
18        .attr("d", this.line(this.stationData));
19
20      svg.selectAll(".line1")
21        .attr("d", this.line1(this.moveAverage))
22        .style("opacity", 1);
23
24      svg.selectAll(".line2")
25        .attr("d", this.line2(this.cutMoveAverageDay))
26        .style("opacity", 1);
27
28      svg.select(".axis.axis--x") // change the x axis
29        .duration(750)
30        .call(this.xAxis);
31
32      svg.select(".axis.axis--y") // change the y axis
33        .duration(750)
34        .call(this.yAxis);
35    }
```

15). Destroy: Unsubscribe

```
1   ngOnDestroy() {
2       if (this.subscription) {
3         this.subscription.unsubscribe();
4       }
5     }
```