In Class 06 – Report – Group 05

Server was built using Node JS and PHP for database. Database contains a table named “data” with below fields:

Field Name	Data Type
id	integer
cost	float
Sales	float
item	string

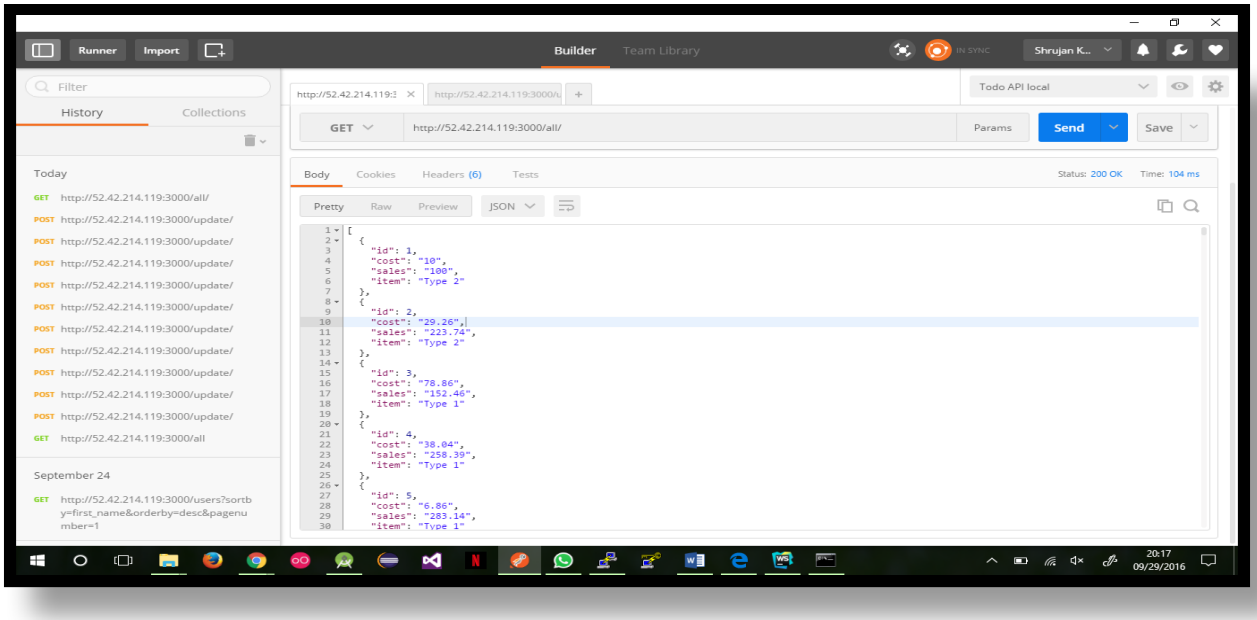
API :

1) To get data of all the points

URL : **http://52.42.214.119:3000/all**

Method : GET

Returns : JSON format data of the data points

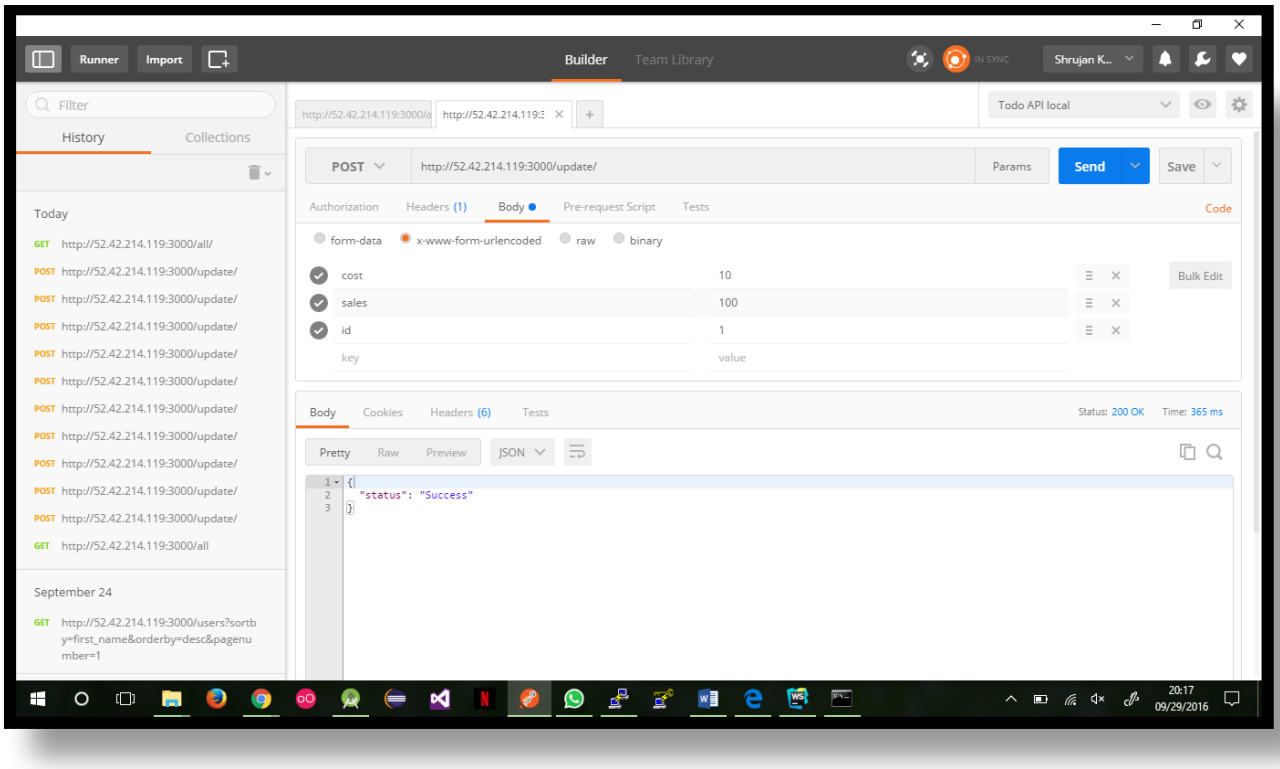


2) To update data of the moved point

URL : **http://52.42.214.119:3000/update/**

Method : POST

Parameters : cost , sales , id



Algorithm used in implementing the chart:

All the data points records are read from the first API and are stored into an Arraylist. Later this array list is passed into a method [getDataOfType()] which filters the data based on the parameter passed. This filtered data is sorted further and plotted on the Scatter chart. For each type of category different color is assigned as shown below.

```
ScatterDataSet type1 = new ScatterDataSet(getDataOfType("Type 1",dataPoints), "Type 1");
type1.setScatterShape(ScatterChart.ScatterShape.CIRCLE);
type1.setScatterShapeHoleColor(Color.RED);
type1.setScatterShapeHoleRadius(3f);
type1.setColor(Color.RED);
ScatterDataSet type2 = new ScatterDataSet(getDataOfType("Type 2",dataPoints), "Type 2");
type2.setScatterShape(ScatterChart.ScatterShape.CIRCLE);
type2.setScatterShapeHoleColor(Color.BLUE);
type2.setScatterShapeHoleRadius(3f);
type2.setColor(Color.BLUE);
ScatterDataSet type3 = new ScatterDataSet(getDataOfType("Type 3",dataPoints), "Type 3");
type3.setScatterShape(ScatterChart.ScatterShape.CIRCLE);
type3.setScatterShapeHoleColor(Color.GREEN);
```

Method – getDataOfType()

```
private ArrayList<Entry> getDataOfType(String s, ArrayList<Data> dataPoints) {
    ArrayList<Entry> matching=new ArrayList<Entry>();
    switch(s){
        case "Type 1":
                for(Data d:dataPoints){
                    if(d.getItem().equals("Type 1"))
                        matching.add(new Entry(Float.parseFloat(d.getCost()),Float.parseFloat(d.getSales())));
                }

            break;
        case "Type 2":
            for(Data d:dataPoints){
                if(d.getItem().equals("Type 2"))
                    matching.add(new Entry(Float.parseFloat(d.getCost()),Float.parseFloat(d.getSales())));
            }
            break;
        case "Type 3":
            for(Data d:dataPoints){
                if(d.getItem().equals("Type 3"))
                    matching.add(new Entry(Float.parseFloat(d.getCost()),Float.parseFloat(d.getSales())));
            }
            break;
    }
    Collections.sort(matching, new EntryXComparator());

    return matching;
```

Algorithm used in implementing the drag feature:

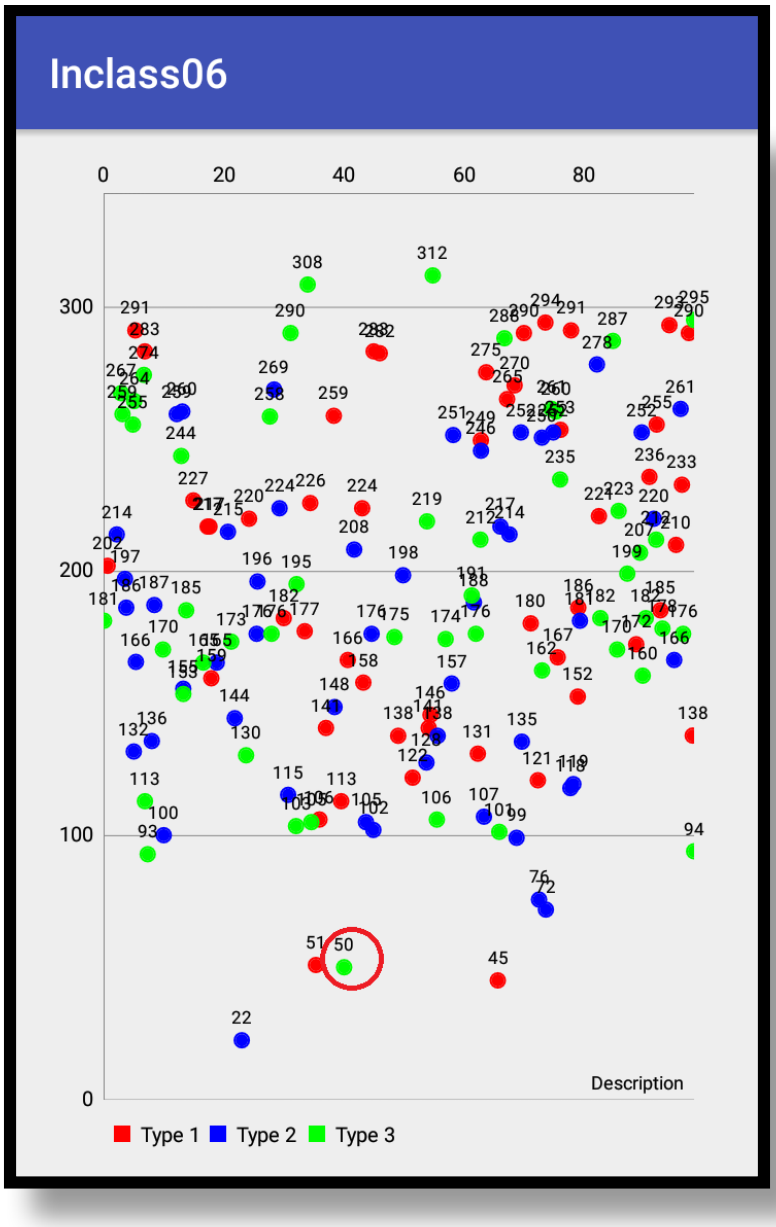
For drag feature the scatter chart is set to OnChartGestureListener and with the help of onChartGestureStart() and onChartGestureEnd() position of initial touch and drop is captured. With the intial touch axis points the id of the data point is determined from the stored list and passed the new data with id to the server for update.

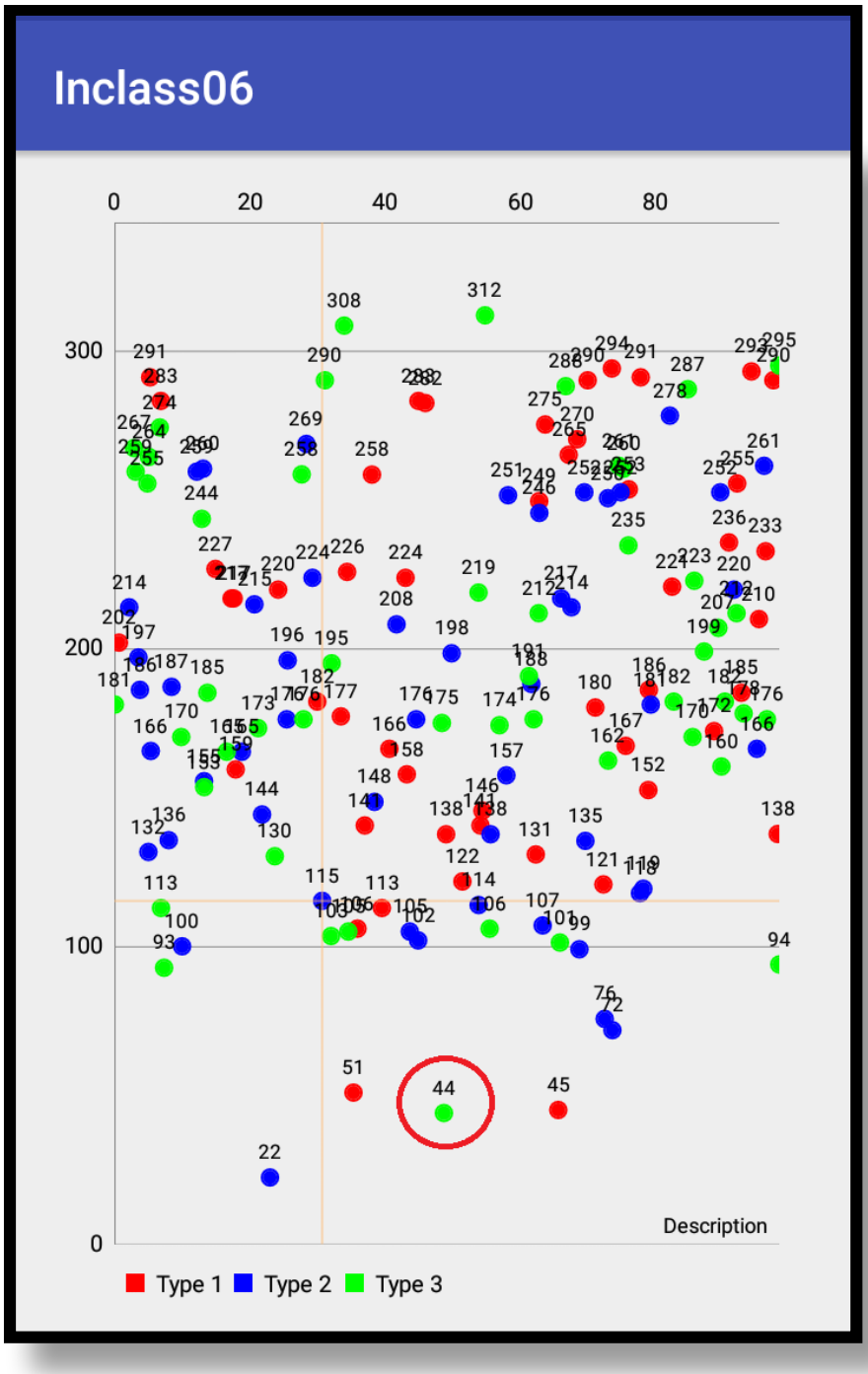
```
public void onChartGestureEnd(MotionEvent me, ChartTouchListener.ChartGesture lastPerformedGesture) {
    if (he != null) {
        Entry e = mChart.getScatterData().getEntryForHighlight(he);
        for (Data d : finalData) {
            if (d.getCost().equals("" + e.getX()) && d.getSales().equals("" + e.getY())) {
                MPPointD values = mChart.getValuesByTouchPoint(me.getX(), me.getY(), YAxis.AxisDependency.LEFT);

            try {
                    JSONObject json1 = new JSONObject();
                    json1.put("id", d.getId());
                    json1.put("cost", values.x+"" );
                    json1.put("sales", values.y+"");
                    Log.d("data",json1.toString());
                    StringEntity se = new StringEntity(json1.toString());
                    se.setContentType(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));
                    new AsyncTaskAPI().execute(se);
                } catch (JSONException e1) {
                    e1.printStackTrace();
                } catch (UnsupportedEncodingException e1) {
                    e1.printStackTrace();
                }
            }
        }
```

Application screen shots:

In the first screen shot data point with value 50 has been moved to another position value 40 as shown in second figure.




Inclass06
0
20
40
60
80
300
200
100
0
Description
Type 1
Type 2
Type 3