2022-2023 第 2 学期

《互联网开发-Web 框架与应用》

课程大作业报告

班级： 21 级计算机科学与技术 2 班　姓名：马菁含　　学号： 32021010069

〇、需要额外说明的问题

我完成了两个功能、除了完成本课程规定的实现地铁线路小车动态到站时间以及离线时间外，我额外增加了一个地铁按照不同线路显示在地画布上的功能。采用了 bulma.min.css 开源框架，改变按钮以及其他表单标签的样式，增加了网站的美观性，我还采用了 saveSvgAsPng.js 和 svg-pan-zoom.min.js 在 GitHub 上开源的 JavaScript 库，用于通过鼠标滚轮或拖动的方式实现 SVG 画布图像的平移和缩放功能，并且能将 SVG 画布图像保存为 PNG 文件至本地相册。在打开网站时，只需要点击增加路线按钮即可增加不同的地铁线路，并且可以自由选择起始位置。增加完线路之后，在下方的小车动态到站显示模块即可出现小车环形地铁二号线的动态过程。

一、功能概述

本系统是一个基于 jQuery.js、svg-pan-zoom.min.js 和 saveSvgAsPng.js 框架的 Web 应用程序，它实现了动态显示地铁线路的小车到站时间和离站时间，并支持按照不同线路在地图上显示地铁路线的功能。用户可以通过增加路线按钮添加不同的地铁线路，并可以自由选择起始位置。一旦添加完线路，用户就可以在下方的小车动态到站显示模块中看到小车在环形地铁二号线上的动态过程。

此外，本系统还提供了保存生成的地铁图片到本地的功能。如果用户无法下载生成的图片，只需点击生成图片按钮即可将图片显示在下方。整个系统的目标是提供一个方便、实用和易于使用的地铁线路动态显示应用程序，以满足用户的需求。

二、实现原理和方法

本系统采用了 jQuery 和 SVG 技术来实现动态显示地铁线路的小车到站时间和离站时间，以及按照不同线路在地图上显示地铁路线的功能。具体原理和方法如下：

jQuery 技术：jQuery.js 是一种基于 JavaScript 的轻量级的开源 JS 库，它简化了 HTML 文档的遍历、事件处理、动画效果和 AJAX 交互等操作，使得开发者可以更加方便快捷地开发 Web 应用程序。在本系统中，我们使用 jQuery 库来实现页面元素的动态显示和交互，例如添加路线按钮、小车动态到站显示模块等。

SVG 技术：SVG 是一种基于 XML 的矢量图形格式，它可以实现高质量的图形显示效果，并且支持动态交互和动画效果。在本系统中，我们使用 SVG 技术来显示地铁线路和相关的图形元素，例如地铁车站、小车、路线等。具体实现方法是使用 SVG 元素和相关属性来绘制图形，并使用 JavaScript 来控制图形的动态效果和交互行为。

saveSvgAsPng.js 技术：saveSvgAsPng.js 是一个 JavaScript 库，它可以将 SVG 图形保存为 PNG 格式的图像文件。在本系统中，我们使用 saveSvgAsPng.js 来实现将生成的地铁图片保存到本地的功能。具体实现方法是在生成图片按钮点击事件中调用 saveSvgAsPng.js 库的相关函数，将 SVG 图形转换为 PNG 格式并保存到本地。

svg-pan-zoom.min.js 技术：svg-pan-zoom.min.js 是一个 JavaScript 库，它提供了 SVG 图形的缩放、平移等交互操作。在本系统中，我们使用 svg-pan-zoom.min.js 技术来让用户可以放大、缩小和平移地图，以便更好地查看地铁线路和相关的图形元素。具体实现方法是在地图区域添加 svg-pan-zoom.min.js 库的相关属性和事件处理程序，以实现交互效果。

三、实验和测试环境

本系统开发和测试环境如下：

表 1 软硬件环境

软硬件环境	软硬件参数
CPU	Intel(R) Core(TM) i5-7300U CPU @

	2.60GHz　2.71 GHz
内存	8.00 GB
操作系统	Windows NT x64 6.1.7601
集成开发环境	VS Code 1.62.2
浏览器	Chrome 101.0.4951.54
服务器	http-server
软件环境	jQuery 1.7.3
	jQuery JavaScript Library v3.6.3
	saveSvgAsPng.js
	svg-pan-zoom.min.js
	Vue.js 2.7

四、运行和测试结果

(一). 北京地铁线路图绘制部分

1．用户可以通过点击“增加路线”按钮，添加新的地铁线路。如图 1，用户需要选择新线路的起点和终点站点，系统会自动在地图上按照选定的起点和终点站点展示所选的全部线路。

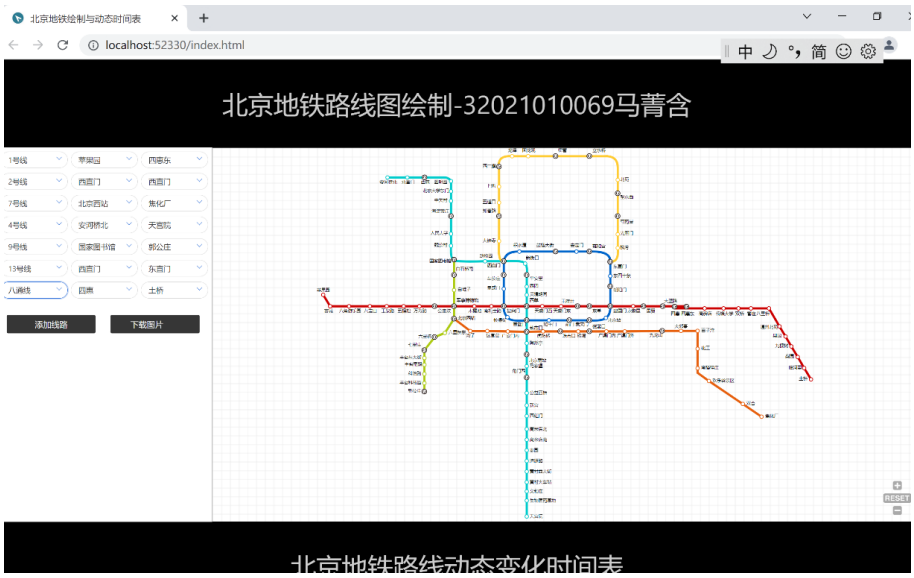

图 1 地铁线路展示效果

2. 当用户选择的起点和终点站点之间需要换乘时，可以通过选择选择不同线路的起点与终点，绘制出一条换成的路线，如图 2：由北京地铁十号线首经贸地铁站-换乘地铁四号线-宣武门-换成地铁二号线-北京站。

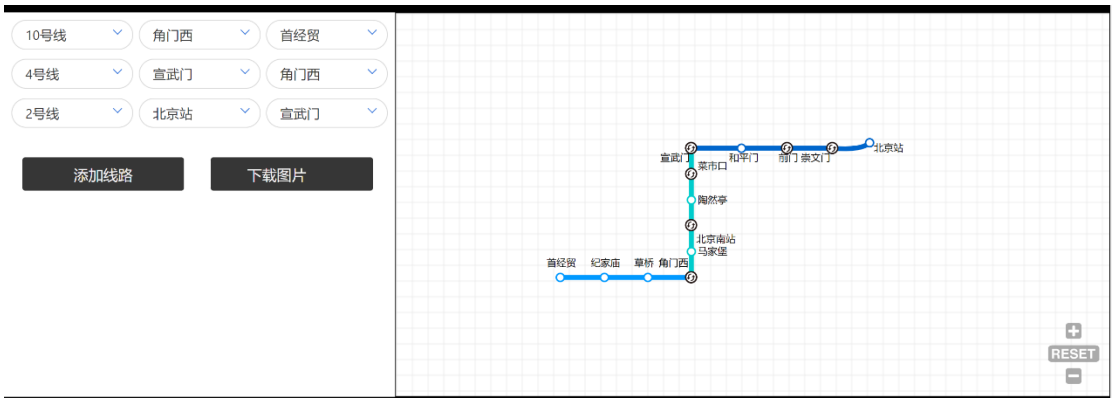

图 2 需换乘的路线展示图

3. 如图 3 所示，用户可以通过鼠标滚轮滑动、鼠标拖拽、以及在画布右下方的“+”、“-”、“reset”按钮来自由地浏览地铁线路和站点信息。这些操作可以让用户更加方便地查看地铁线路和站点信息，同时也提高了用户体验的舒适度。

图 3 自由平移与缩放地铁线路图画布

4. 添加了下载图片的选项，用户可以点击下载图片按钮来将生成的地铁线路图保存到本地。如图 4 所示。

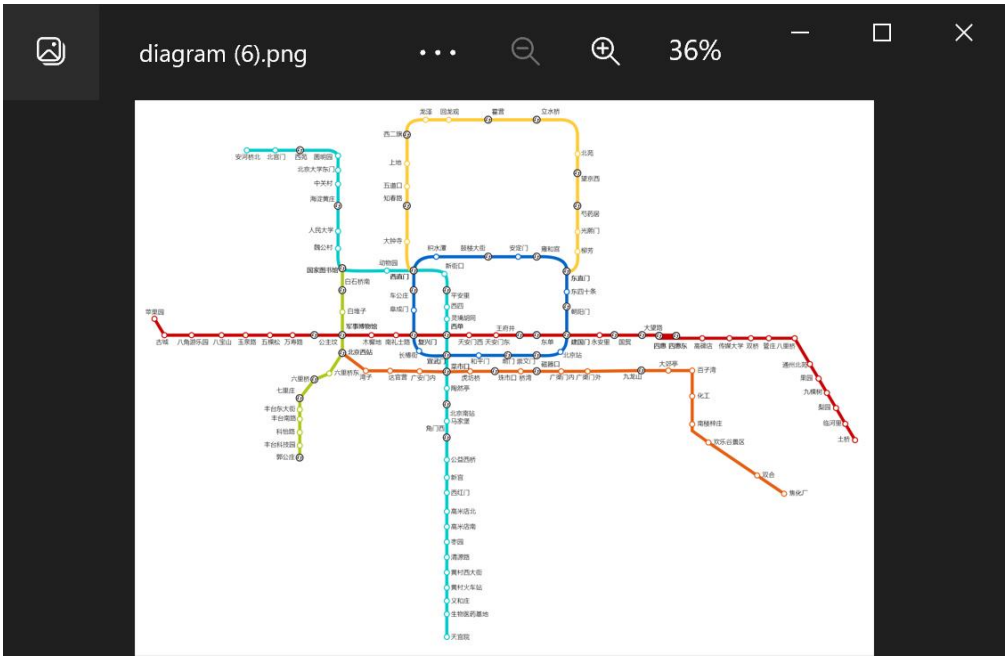

图 4 生成的 PNG 图片

5. 实现了生成图片的功能，用户可以切换下载图片按钮变成生成图片按钮(如图 5)，来生成图片，并将其显示在该地铁线路图下方（如图六）。这使得用户可以更方便地分享和保存地铁线路图。

图 5 切换至生成图片模式

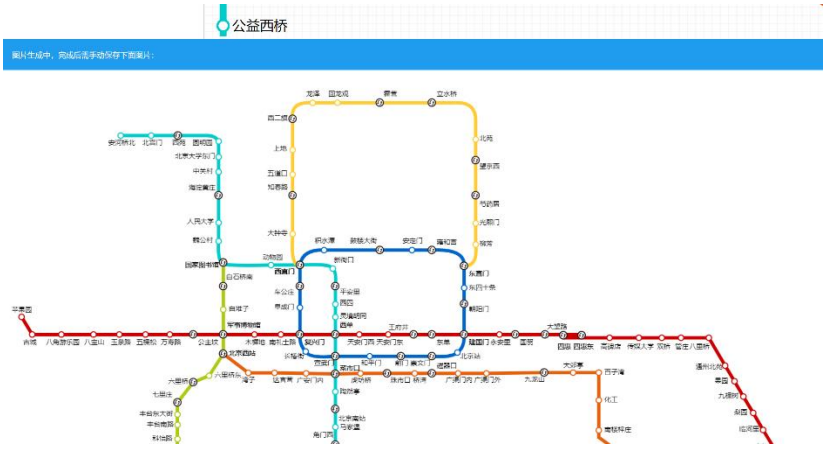

图 6 生成图片至画布下方

（二）. 北京地铁线路图动态变化

使用了定时器 setInterval() 来定时更新小车的位置和时间。此外，我还通过初始化停车标志数组和时间以及列车编号来管理小车的位置和状态。在更新列车位置的过程中，我先判断是否到达下一个停靠站点，如果是，则更新停车标志并判断是否需要移除列车。如果列车已经到达终点，则跳过处理。最后，计算列车位置的坐标并更新 SVG 元素的位置。通过以上方式，如图 7 所示，我实现了模拟一辆小车在二号线上的运行情况，并且实现了小车在线路上的动态到站与离站功能，可以让用户更加直观地了解小车的运行情况。

图 7 小车运行动态图

```
8:7:59
new train 0 start at 7:0:1
new train 1 start at 7:3:0
new train 2 start at 7:6:0
new train 3 start at 7:9:0
new train 4 start at 7:13:0
new train 5 start at 7:16:0
new train 6 start at 7:19:0
new train 7 start at 7:22:0
new train 8 start at 7:25:0
```

图 8 小车运行动态日志

五、结论和总结

1. 总结

此项大作业最终实现了北京地铁线路图的绘制和动态变化，用户可以通过点

击按钮、鼠标拖拽等方式自由浏览地铁线路和站点信息，并且实现了添加新线路、换乘路线、下载图片、生成图片等功能。同时，通过使用定时器和 SVG 元素的操作，实现了小车在地铁线路上的模拟运行和到站离站的动态效果。

2.不足和问题：

在操作上，交互性不强，例如用户想要查看其他路线的小车动态运行情况无法实现，只涉及了一条地铁 2 号线环线的动态小车运行情况。在代码实现上，可能存在性能问题，尤其是在绘制地铁线路图和动态变化小车位置时，需要处理大量的数据，可能会导致系统卡顿或出现延迟。同时，对于大规模的地铁线路图，需要更高效的算法和数据结构来优化系统的性能。有无法解决的问题，比如小车会在上下两个地图的模块都显示，并没有得到解决。

3.改进方案：

可以增加更多的交互性和可视化效果，如使用更多的动画效果来提升用户体验；可以优化代码结构和算法，采用更高效的数据结构和算法来提高系统性能。同时，使用改变时间表数组，让其循环来达到小车无限循环运行的目的。

附：关键代码

```
// 遍历数据（DATA）数组，将线路名字和站点名字存储到 lines 和 line_stations 数组中
for (var i = 0; i < DATA.length; i++) {
    lines.push(DATA[i]['name']);
    var temp = [];
    var stations = DATA[i]['stations'];
    for (var j = 0; j < stations.length; j++) {
        temp.push(stations[j]['name']);
    }
    line_stations.push(temp);
}
// 创建线路选择下拉框
    var select_line = create_select_tag("select_line_" + select_count, lines, "line", function () {
        var pos = this.value;
        var stations = line_stations[pos];
        var start = this.parentNode.nextSibling.SELECT;
        var end = start.parentNode.nextSibling.SELECT;
        start.options.length = 0;
```

```
        end.options.length = 0;
        for (i = 0; i < stations.length; i++) {
            start.options.add(new Option(stations[i], i));
            end.options.add(new Option(stations[i], i));
        }
        end.options[i - 1].selected = true;
        if (!init_flag)
            flush_svg()
            ;

    });

    // 创建起点站选择下拉框
    var select_start_station = create_select_tag("select_station_"
+ select_count, ['---'], 'start', function () {
        var end = this.parentNode.nextSibling.SELECT;
        if (parseInt(end.value) < parseInt(this.value)) {
            alert("终点站在始发站后面");
            return;
        }
        flush_svg()
        ;
    });

    // 创建终点站选择下拉框
    var select_end_station = create_select_tag("select_station_" +
select_count, ['---'], 'end', function () {

        var start = this.parentNode.previousSibling.SELECT;
        if (parseInt(this.value) < parseInt(start.value)) {
            alert("终点站在始发站后面");
            return;
        }
        flush_svg();
    });

// 绘制一条地铁线路
function draw_line(line, start, end) {
    var svg = document.getElementById('map');
    var name = line['name'];
    var color = line['color'];
    var stations = line['stations'];
    var is_reverse = reverse.has(name);
    var first = true;
```

```
    var path_data = '';
    var draw_name_img = '';    for (var i = start; i < end + 1; i++)
{
        var s = stations[i];
        draw_name_img += s['draw_name'];
        draw_name_img += s['draw_img'];
        var x=parseInt(s['xy'].split(',')[0]);
        var y=parseInt(s['xy'].split(',')[1]);
        if ( x< min_x)
            min_x = x;
        if (y < min_y)
            min_y = y;

        if (x > max_x)
            max_x = x;
        if (y > max_y)
            max_y = y;

        if (first) {
            path_data += 'M' + s['xy'];
            first = false;
            continue;
        }
        if (is_reverse) {
            if (s['draw_type'] == 'L') path_data += 'L' + s['xy'];
            path_data += s['draw_args'];
        }
        else {
            path_data += s['draw_args'];
            if (s['draw_type'] == 'L') path_data += 'L' + s['xy'];
        }
    }

    var draw_path = '<path d="' + path_data + '" eletype="1"
fill="none" stroke="' + color + '" stroke-width="8"></path>';
    var draw_group = '<g id="' + name + '">' + draw_path +
draw_name_img + '</g>';

    svg.innerHTML += draw_group;
}

// 刷新 SVG 画布
function flush_svg() {
    var svg = document.getElementById('map');
```

```
    if (zoomSvg != null)
        zoomSvg.destroy();
    svg.innerHTML = '';

    var                          select_data                          =
document.getElementById('select_data').children;
    for (var i = 0; i < select_data.length; i++) {
        var temp = select_data[i].childNodes;
        var select_line = temp[0].SELECT;
        var select_start_station = temp[1].SELECT;
        var select_end_station = temp[2].SELECT;
        draw_line(DATA[parseInt(select_line.value)],
parseInt(select_start_station.value),
parseInt(select_end_station.value));
    }
    svg_data = svg.innerHTML;

    zoomSvg = svgPanZoom('#map', {
        zoomEnabled: true,
        controlIconsEnabled: true,
        fit: true,
        center: true,
        zoomScaleSensitivity: 0.5
    });
}

// 获取切换按钮元素并添加点击事件，点击后更改下载类型并显示按钮文本
var oBtn3 = document.getElementById('change');
oBtn3.onclick = function () {
download_type = !download_type;
if (download_type)
oBtn2.innerHTML = '<span style="padding-left: 10px">下载图片</span>'
else
oBtn2.innerHTML = '<span style="padding-left: 10px">生成图片</span>';
};

// 使用 svgPanZoom 插件对 SVG 进行缩放和拖动操作
zoomSvg = svgPanZoom('#map', {
zoomEnabled: true,
controlIconsEnabled: true,
fit: false,
center: true,
zoomScaleSensitivity: 0.5
});
```

```
// 调用 flush_svg 函数对 SVG 元素进行初始化
flush_svg();
init_flag = false; // 初始化标志位

        //定时器，每隔 10 毫秒执行一次
        setInterval(function () {
            //增加时间
            time = time + 0.00032
            //更新显示时间的文本
            $("#time").text(timeconv(time))

            //判断是否需要添加新的列车
            if (train < 24 && time > str2time(timetb[0][train])) {
                //创建圆形 SVG 元素并添加到文档中
                xml = jQuery.parseXML('<circle id="yz-' + train + '"
xmlns="http://www.w3.org/2000/svg" cx="' + ln1[0].x + '" cy="' +
ln1[0].y + '" r="10" stroke="black" stroke-width="2" fill="red"/>');
                $("g").append(xml.documentElement)
                //添加操作记录到日志中
                $("#log").append("<p>"+"new train " + train + " start
at " + timeconv(time) +"</p>")
                //增加列车编号
                train = train + 1
            }

            //更新列车的位置
            for (i = 0; i < train; i++) {
                //判断是否到达下一个停靠站点
                if (time > str2time(timetb[stop[i]][i])) {
                    //更新停车标志，判断是否需要移除列车
                    stop[i] = stop[i] + 1
                    if(timetb[stop[i]][i] == ""){
                        $("#yz-" + i).remove()
                        $("#log").append("<p>"+"remove train " + i +"
at " + timeconv(time) + "</p>")
                    }
                }
                //如果列车已经到达终点，则跳过
                if (stop[i]>26){
                    continue
                }

                //计算列车位置的坐标
```

```
            count = (time - str2time(timetb[stop[i] - 1][i])) /
(str2time(timetb[stop[i]][i]) - str2time(timetb[stop[i] - 1][i]))
            tx = ln1[stop[i] - 1].x * (1 - count) + ln1[stop[i]].x
* count
            ty = ln1[stop[i] - 1].y * (1 - count) + ln1[stop[i]].y
* count
            //更新 SVG 元素的位置
            $("#yz-" + i).attr("cx", tx)
            $("#yz-" + i).attr("cy", ty)
        }
    }, 10)

})
```