Applied Research Report:

Using Chart.js for Data Visualization in a PHP + MySQL Web Portal

Course: CSIS 4495 Applied Research Project
Author: Hao SUO (300392463)
Project context: “world rebalance” game analytics portal (vehicle usage, win trends, etc.)
Instructor: Bambang Sarif

Abstract:

This report documents my end-to-end learning and implementation of Chart.js for browser-based data visualization, integrated with a PHP + MySQL backend. I first practiced static charts (bar, line) on HTML <canvas>, then built a PHP API to query MySQL and return JSON, and finally used fetch() on the frontend to render dynamic charts. The outcome is a working demo that visualizes real database values with responsive, styled charts suitable for inclusion in the project portal.

Objectives:

1. Learn the Chart.js mental model: type → data → options, rendered onto an HTML5 <canvas>.
2. Implement Bar and Line charts with static data to master the API surface.
3. Build a PHP endpoint that queries MySQL and outputs JSON in a shape directly consumable by Chart.js.
4. Use fetch() to load server data and render/update charts in the browser.
5. Document pitfalls, troubleshooting steps, and next steps for production readiness.

Concepts Learned:

1 The <canvas> Role

- <canvas> is a blank drawing surface. It does not know chart types.
- Chart type is defined in JavaScript when creating new Chart(canvas, config).

2 Chart.js Configuration

- type: chart kind ('bar', 'line', 'pie'.....).
- data: what to draw (X-axis labels, one or more datasets with numeric arrays).
- options: how to draw (title, legend, scales, tooltips, responsiveness, interactivity).

Below is a structure that can make a simple barChart:

```
vehicleLabels = ['APC', 'Tank', 'Jeep', 'Artillery', 'Bike', 'Drone'];
let vehicleCounts = [12, 25, 9, 6, 15, 4];

const barchart = new Chart(
    document.getElementById('barChart'),
    //this is the second parameter of new Chart
    {
```

```
            type: 'bar',    // tell Chart.js to draw a bar chart
            data: {             // chart data
                labels: vehicleLabels,      // X-axis names
                datasets: [                         // what to draw
                    {
                        label: "Used Times",    // chart legend name
                        data: vehicleCounts,    // real data
                        borderWidth: 1,             // bar border width
                        // if there are many labels, Chart.js will set colors auto
                        // backgroundColor = fill color
                        // borderColor = line color
                    },
                    // if you have more data group, add more {}
                    {
                        label: 'Test',
                        data: [1, 2, 3, 4, 5, 6],
                        borderWidth: 1
                    }
                ]
            },

            // options tell chart HOW to draw
            options: {
                responsive: true,                   // make chart auto fit container size
                maintainAspectRatio: false, // can change height freely (not fixed)
                plugins: {
                    title: { display: true, text: 'Vehicle Used Times (Example)' }, // chart title
                    legend: { display: true }    // show or hide the legend
                },
                interaction: { mode: 'nearest', intersect: true }, // when mouse move close,
show tooltip
                scales: {
                    x: { grid: { display: false } }, // hide x-axis grid line
                    y: {
                        beginAtZero: true,      // y-axis start from 0
                        ticks: { stepSize: 5 }    // each tick step = 5 (like 0,5,10,15...)
                    }
                }
            }
        }
);
```

3 Minimal HTML

We need a test html (div) as a placeholder to draw the barChart afterwards

```html
<html lang="en">
<head>
    <style>
        canvas {
        }

        /* Do not add padding to canvas to avoid size calculation issues */
    </style>
    <title>Chart Test Page</title>
</head>

<body>
    <div class="wrap">
        <div class="row">
            <div class="col">
                <h3>Bar Chart: Vehicle Usage Count</h3>
                <div class="chartbox">
                    <canvas id="barChart"></canvas>
                </div>
            </div>

            <!-- lineChart test -->
            <div class="col">
                <h3>Line Chart: Vehicle Usage Count</h3>
                <div class="lineChartBox">
                    <canvas id="lineChart"></canvas>
                </div>
            </div>

        </div>

    </div>

</body>
```

for barChart

for lineChart as a placeholder

4 Line Chart (Static)

```javascript
const vehicleLabels = ['APC', 'Tank', 'Jeep', 'Artillery', 'Bike', 'Drone'];
let vehicleCounts1 = [12, 25, 9, 6, 15, 4];

const lineChart = new Chart(
  document.getElementById('lineChart'),
  {
    type: 'line',
    data: {
      labels: vehicleLabels,
      datasets: [
        {
          label: 'count of using',
          data: vehicleCounts1,
          fill: false,
          tension: 0.2, //
          pointRadius: 3  // size of point
        }
      ]
    },
    options: {
      responsive: true,
      maintainAspectRatio: false,
      plugins: {
        title: { display: true, text: 'legend' },
        legend: { display: true },
        tooltip: { enabled: true }
      },
      scales: {
        x: { grid: { display: false } },
        y: { beginAtZero: true }
      }
    }
  }
);
```

Key point:

- Kept labels.length === data.length.
- Avoided padding on <canvas>; used a parent with fixed height + maintainAspectRatio:false.
- Verified script order: Chart.js before my scripts.

Interaction with mysql and php:

1 mysql table

```
MariaDB [test]> select * from vehicles;
+----+------------+-------+
| id | name       | count |
+----+------------+-------+
|  1 | APC        |    12 |
|  2 | Tank       |    25 |
|  3 | Jeep       |     9 |
|  4 | Artillery  |     6 |
|  5 | Bike       |    15 |
|  6 | Drone      |     4 |
+----+------------+-------+
6 rows in set (0.000 sec)

MariaDB [test]> desc vehicles
    -> ;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int(11)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(50)  | YES  |     | NULL    |                |
| count | int(11)      | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
3 rows in set (0.016 sec)

MariaDB [test]>
```

2 PHP(getVehicles.php)

This PHP page will send a request to the MySQL database and then handle the response data as JSON.

```
CHART_TEST
barChart1.js
dynamicBarChart.js
getVehicles.php
index.html
index1.html
lineChart1.js
main.js
```

getVehicles.php > ...

```php
<?php
header('Content-Type: application/json');

//连接数据库
$host = "localhost";
$user = "root";
$pass = "";
$dbname = "test";
$conn = new mysqli($host, $user, $pass, $dbname);
if ($conn->connect_error) {
    die(json_encode(["error" => "DB connection failed: " . $conn->connect_error]));
}

//查询数据
$sql = "SELECT name, count FROM vehicles";
$result = $conn->query($sql);   //获得查询结果

$labels = [];
$data = [];

while ($row = $result->fetch_assoc()) {
    $labels[] = $row["name"];    //每一行的name放入到$labels[]数组中
    $data[] = (int)$row["count"];   //每一行的count放入到$data[]数组中
}

//输出 JSON 格式
echo json_encode([      //使用的是关联数组，也就是labels对应$labels,data对应$data
    "labels" => $labels,
    "data" => $data
]);

$conn->close();
?>
```

What I learned:

- fetch_assoc() retrieves each row as an associative array (["name"=>"APC", "count"=>12]), looped into two arrays for Chart.js.
- json_encode() converts PHP arrays to JSON for the frontend.

3 Frontend Fetch(dynamicBarChart.js)

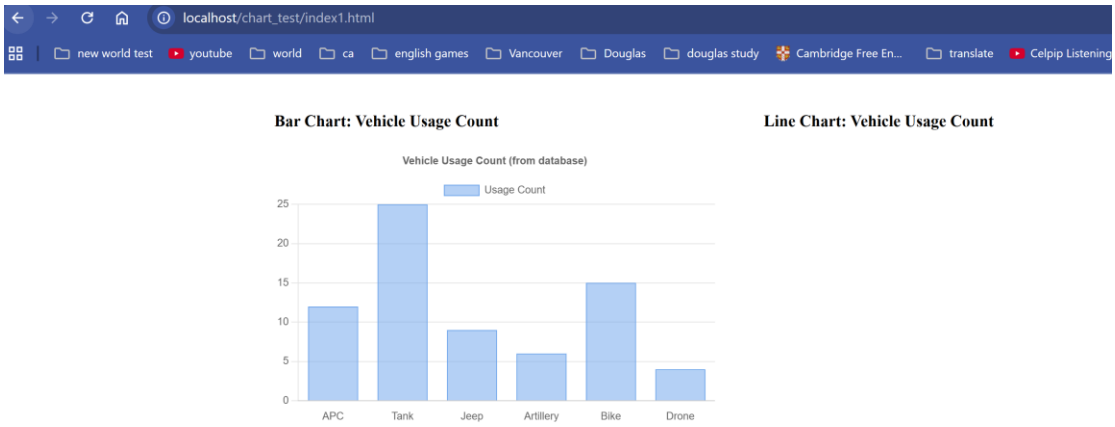
```
CHART_TEST
barChart1.js
dynamicBarChart.js
getVehicles.php
index.html
index1.html
lineChart1.js
main.js

dynamicBarChart.js > then() callback > data > datasets > borderWidth
fetch('getVehicles.php')    // This line runs the getVehicles.php script, and fetch sends a GET request to the backend
  .then(response => response.json())   // Convert the response to a JSON object
  .then(json => {
    const ctx = document.getElementById('barChart');
    new Chart(ctx, {
      type: 'bar',
      data: {
        labels: json.labels,    // "json" here is the whole response, and "labels" comes from it
        datasets: [{
          label: 'Usage Count',
          data: json.data,     // Use "data" from the JSON structure
          backgroundColor: 'rgba(54,162,235,0.5)',
          borderColor: 'rgba(54,162,235,1)',
          borderWidth: 1
        }]
      },
      options: {
        responsive: true,
        maintainAspectRatio: false,
        plugins: {
          title: { display: true, text: 'Vehicle Usage Count (from database)' },
          legend: { display: true }
        },
        scales: {
          x: { grid: { display: false } },
          y: { beginAtZero: true }
        }
      }
    });
  })
  .catch(err => console.error("Failed to load data:", err));
```

What I learned:

- The API must return pure JSON.

Results:



Future Work

- Interactive filters (date ranges, factions, vehicles) to refresh datasets via query params.
- Multiple charts dashboard with coordinated updates.
- Server-side caching (e.g., query memoization) for heavy analytics.
- Additional chart types (doughnut for win share, radar for class performance).

Conclusion:

I have successfully learned and implemented Chart.js to visualize both static and database-driven data inside a web portal. I understand the <canvas> role, the Chart.js configuration (type, data, options), and how to dynamically populate charts using PHP + MySQL + fetch(). The working demo proves my learning outcomes and provides a strong foundation for extending the RebalanceStats analytics dashboard.