SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Projektni zadatak iz predmeta

VIZUALIZACIJA PODATAKA

osu! world map

Student: Oliver Andjić, sveučilišni diplomski studiji računarstvo, DRD

Mentor: [Ime i prezime]

U Osijeku, 06.2025.

SADRŽAJ

1.KV1 - Definiranje projektnog zadatka

1.1. Projektni zadatak

Cilj je prikazati podatke o osu igračima po državama na interaktivan način koristeći mapu svijeta.

Naziv zadatka: osu! World map

Opis problema: osu! igrači po državama

Opis zadatka: statistika o osu! igračima

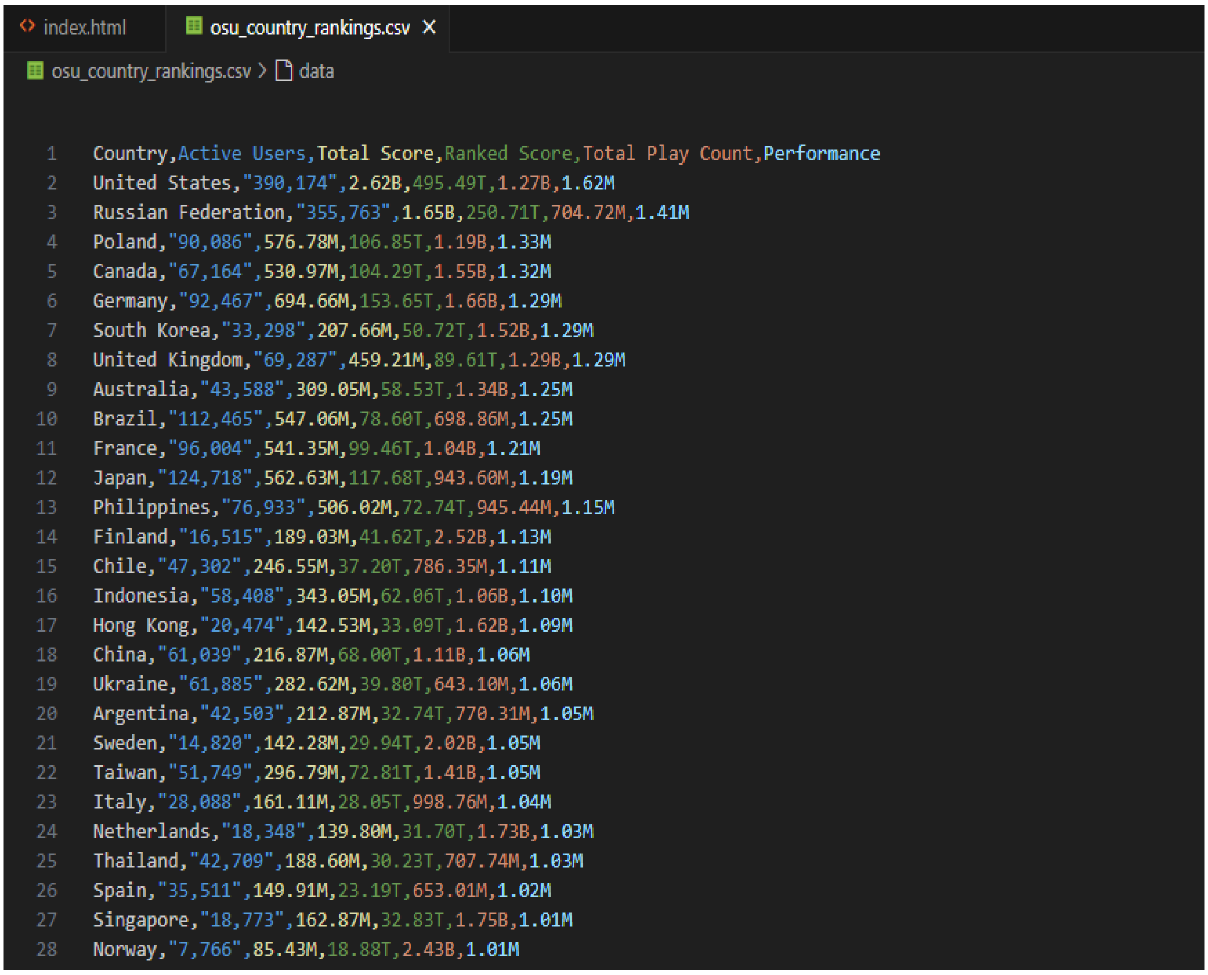
Cilj projekta: interaktivan prikaz statistike o osu! igračima

Poveznica na git repozitorij projekta: https://github.com/PaftDunk22/viz_pod_project

1.2. Podatci

https://osu.ppy.sh/rankings/osu/country

1.3. Obrada podataka

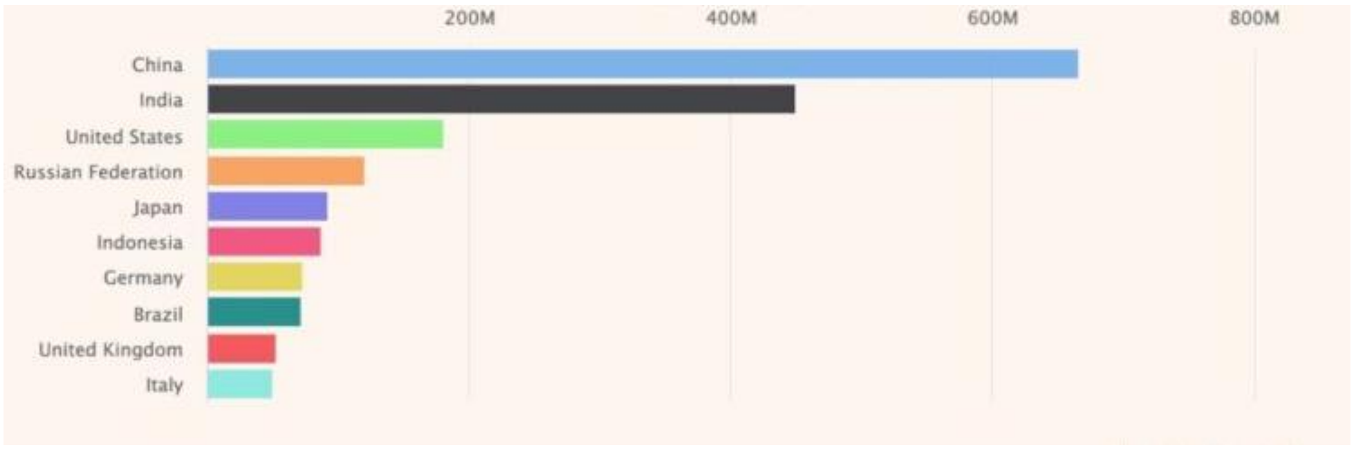


```
Country,Active Users,Total Score,Ranked Score,Total Play Count,Performance
United States,"390,174",2.62B,495.49T,1.27B,1.62M
Russian Federation,"355,763",1.65B,250.71T,704.72M,1.41M
Poland,"90,086",576.78M,106.85T,1.19B,1.33M
Canada,"67,164",530.97M,104.29T,1.55B,1.32M
Germany,"92,467",694.66M,153.65T,1.66B,1.29M
South Korea,"33,298",207.66M,50.72T,1.52B,1.29M
United Kingdom,"69,287",459.21M,89.61T,1.29B,1.29M
Australia,"43,588",309.05M,58.53T,1.34B,1.25M
Brazil,"112,465",547.06M,78.60T,698.86M,1.25M
France,"96,004",541.35M,99.46T,1.04B,1.21M
Japan,"124,718",562.63M,117.68T,943.60M,1.19M
Philippines,"76,933",506.02M,72.74T,945.44M,1.15M
Finland,"16,515",189.03M,41.62T,2.52B,1.13M
Chile,"47,302",246.55M,37.20T,786.35M,1.11M
Indonesia,"58,408",343.05M,62.06T,1.06B,1.10M
Hong Kong,"20,474",142.53M,33.09T,1.62B,1.09M
China,"61,039",216.87M,68.00T,1.11B,1.06M
Ukraine,"61,885",282.62M,39.80T,643.10M,1.06M
Argentina,"42,503",212.87M,32.74T,770.31M,1.05M
Sweden,"14,820",142.28M,29.94T,2.02B,1.05M
Taiwan,"51,749",296.79M,72.81T,1.41B,1.05M
Italy,"28,088",161.11M,28.05T,998.76M,1.04M
Netherlands,"18,348",139.80M,31.70T,1.73B,1.03M
Thailand,"42,709",188.60M,30.23T,707.74M,1.03M
Spain,"35,511",149.91M,23.19T,653.01M,1.02M
Singapore,"18,773",162.87M,32.83T,1.75B,1.01M
Norway,"7,766",85.43M,18.88T,2.43B,1.01M
```

html scraper za pretvorbu podataka o osu igračima sa stranice u csv podaci su aktivni igrači te razni faktori performanse unutar osu! video igre

1.4. Relevantne vrste prikaza za korištene podatke

U ovom projektu će se koristiti mapa svijeta koja će prikazati podatke o osu igračima po državi



Također je moguće za daljni uvid imati i bar chart koji će prikazivati podatke po državama, o primjerice aktivnim korisncima

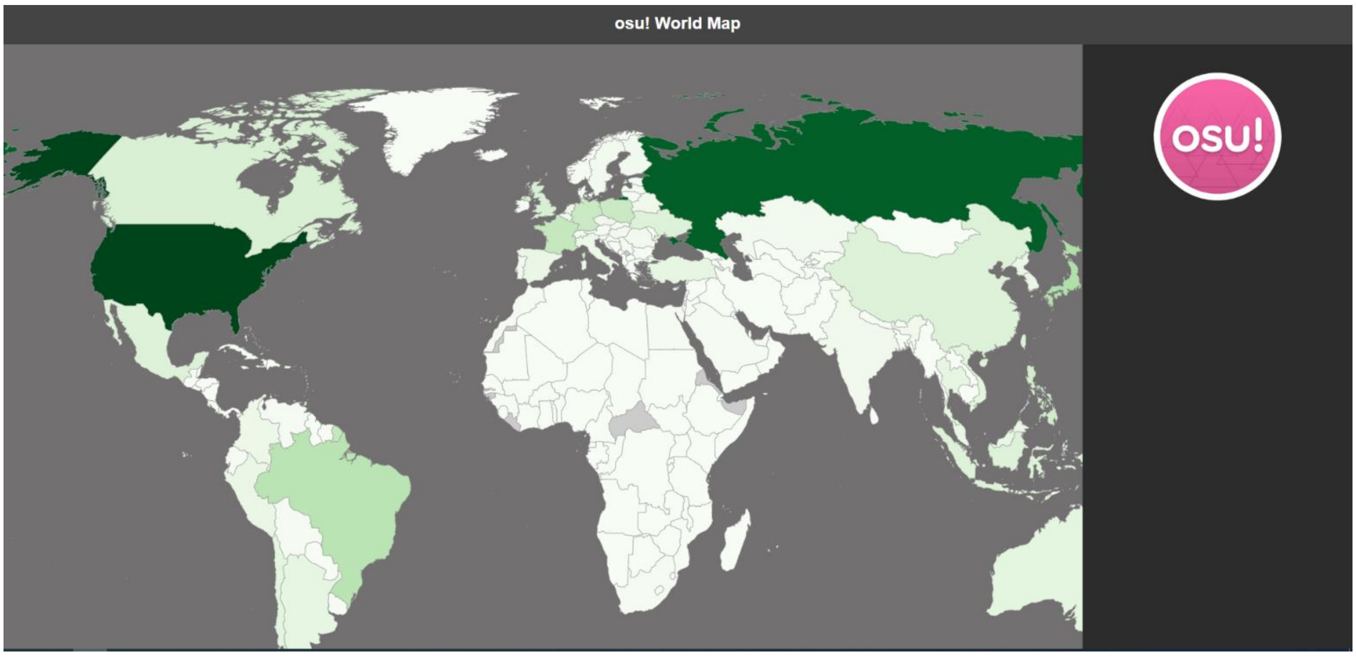
2. KV2 - Dizajn vizualizacije podataka.

2.1. Pitanja na koja vizualizacija daje odgovor

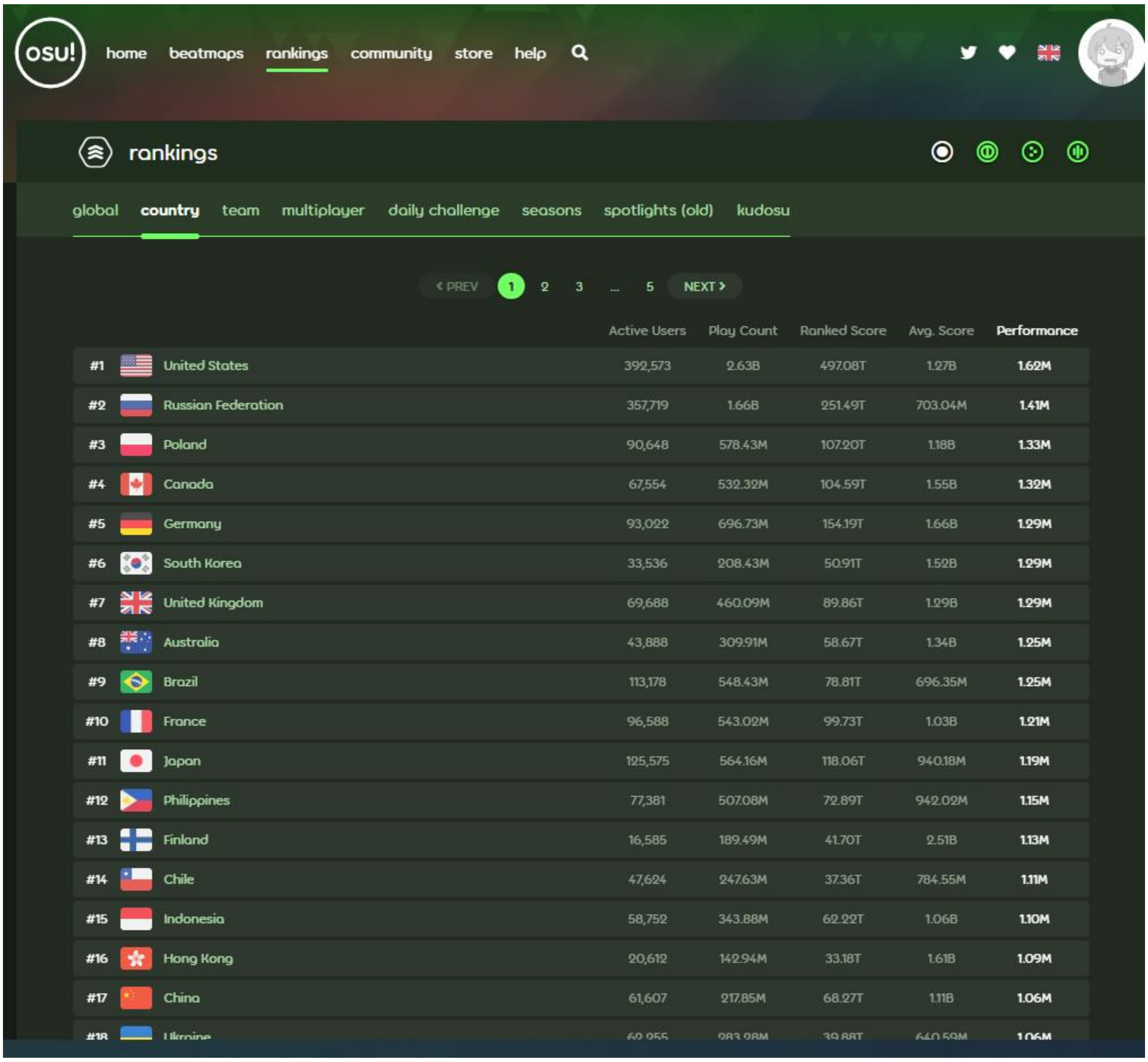
Uvid u podatke za osu igrače po državi.

Primjerice : Koliko aktivnih igrača ima x država?

2.2. Skica vizualizacije podataka



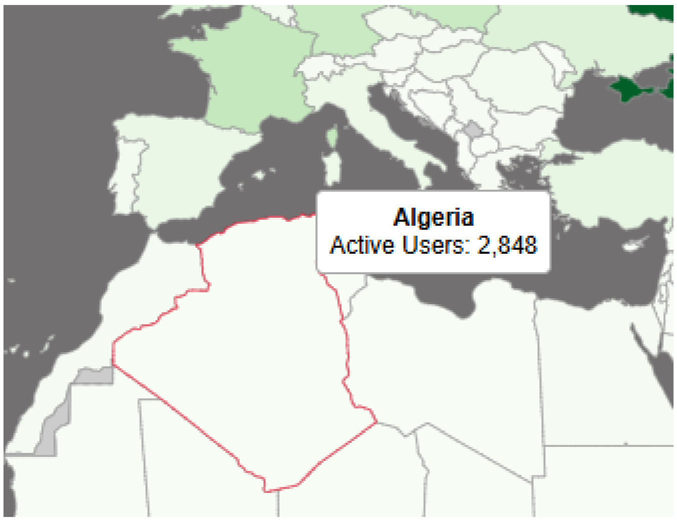
2.3. Postojeća rješenja i primjeri

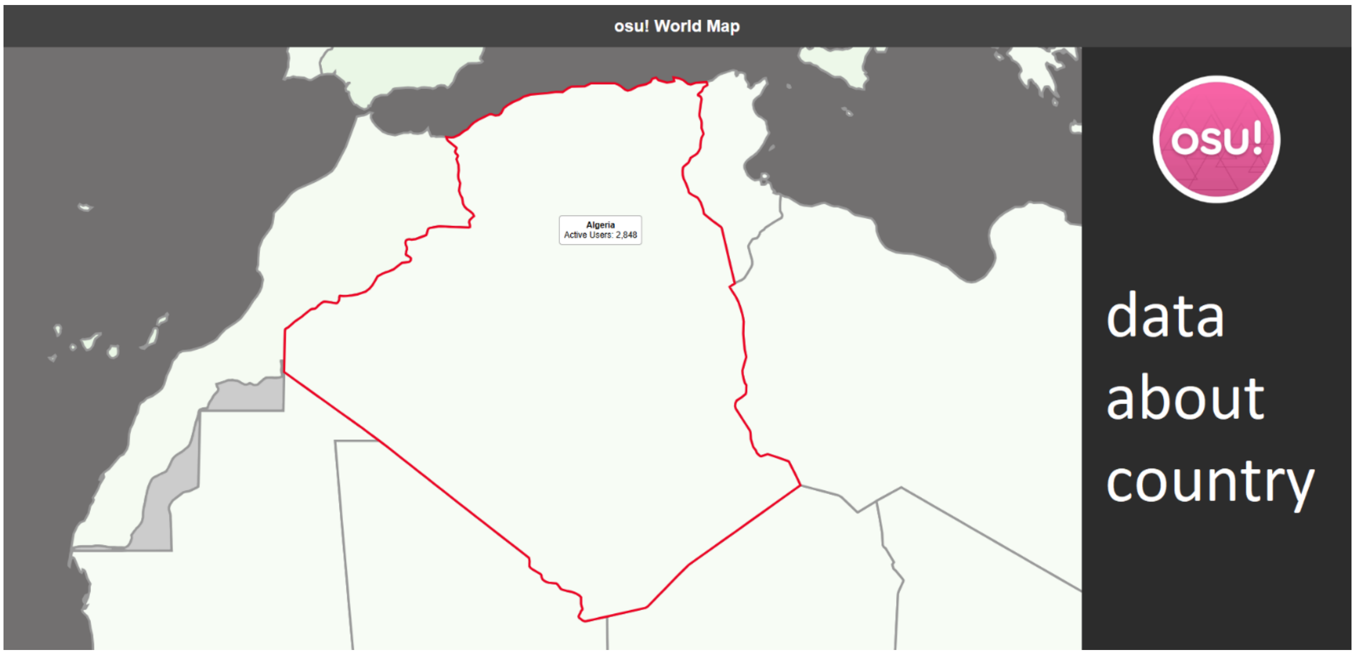


		Active Users	Play Count	Ranked Score	Avg. Score	Performance
#1	United States	392,573	2.63B	497.08T	1.27B	1.62M
#2	Russian Federation	357,719	1.66B	251.49T	703.04M	1.41M
#3	Poland	90,648	578.43M	107.20T	1.18B	1.33M
#4	Canada	67,554	532.32M	104.59T	1.55B	1.32M
#5	Germany	93,022	696.73M	154.19T	1.66B	1.29M
#6	South Korea	33,536	208.43M	50.91T	1.52B	1.29M
#7	United Kingdom	69,688	460.09M	89.86T	1.29B	1.29M
#8	Australia	43,888	309.91M	58.67T	1.34B	1.25M
#9	Brazil	113,178	548.43M	78.81T	696.35M	1.25M
#10	France	96,588	543.02M	99.73T	1.03B	1.21M
#11	Japan	125,575	564.16M	118.06T	940.18M	1.19M
#12	Philippines	77,381	507.08M	72.89T	942.02M	1.15M
#13	Finland	16,585	189.49M	41.70T	2.51B	1.13M
#14	Chile	47,624	247.63M	37.36T	784.55M	1.11M
#15	Indonesia	58,752	343.88M	62.22T	1.06B	1.10M
#16	Hong Kong	20,612	142.94M	33.18T	1.61B	1.09M
#17	China	61,607	217.85M	68.27T	1.11B	1.06M
#18	Ukraine	69,955	283.28M	39.88T	640.59M	1.06M

Prikaz podataka na osu stranici, čisti tekstualni oblik sa državama i raznim podacima o tim državama.

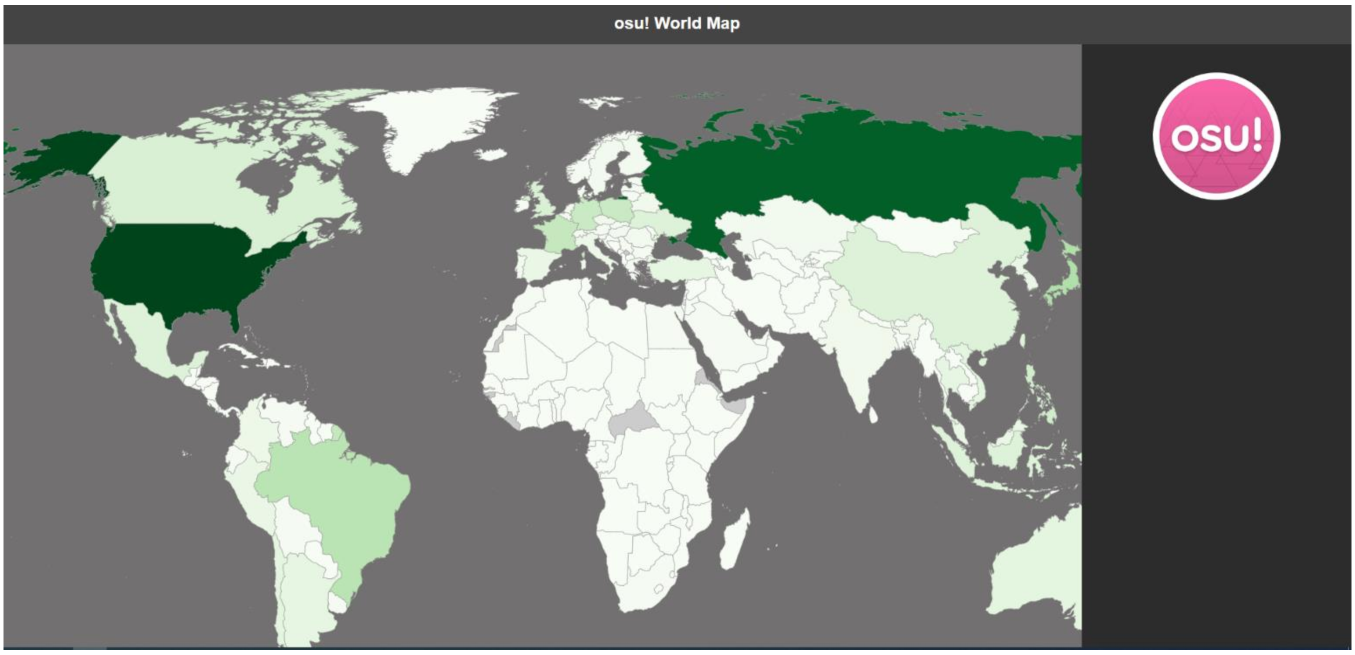
2.4. Prilagodba podataka





Pri highlight-u se prikazuju aktivni igrači, a na klik se zoomira i dobiva veći uvid u statistiku o državi.

2.5. Boje i podatci



Tamnija područja – više igrača
Siva područja – nema podataka

3. KV3 - Izrada prototipne vizualizacije podataka

3.1. Osnovne funkcionalnosti i ponašanja

Prikaz svjetske map te kretanje po mapi.

Zumiranje na države

Prikaz legende za obojane države.

3.2. Napredne funkcionalnosti i ponašanja:

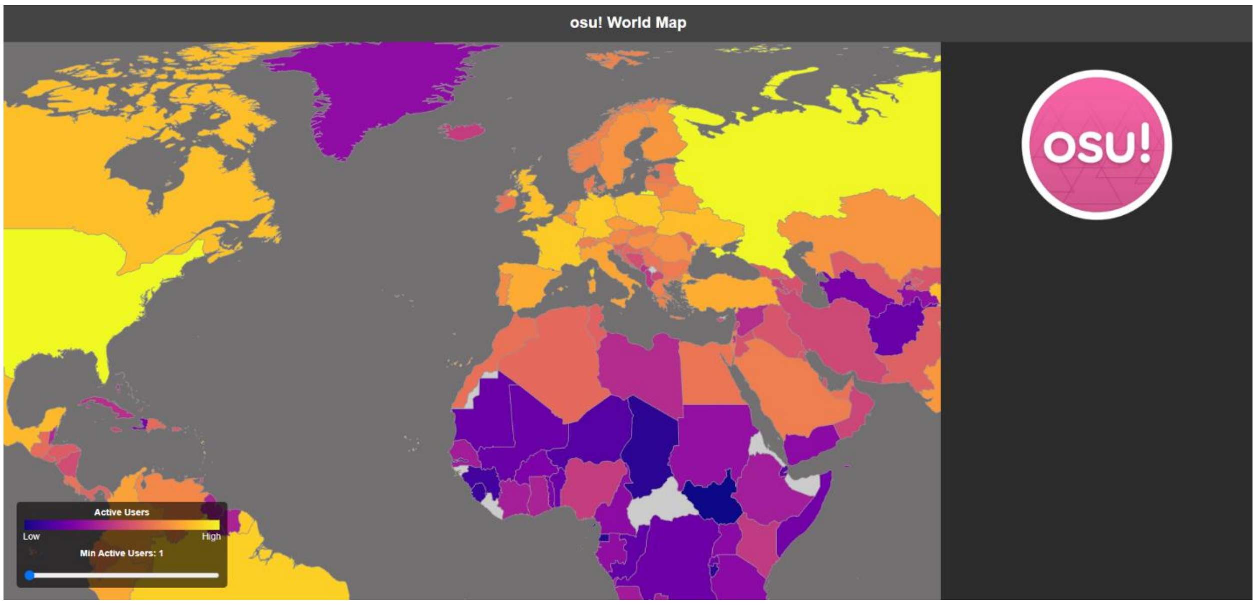
Na klik se prikazuju detaljniji podaci o državi uz zastavu i bar chart koji prikazue podatke o izabranoj državi.

On hover – osnovni podaci o državi te border highlight

Filitriranje mapi ovisno o parametru koji opisuje države.Implementacija osnovnih funkcionalnosti

3.3. Implementacija osnovnog ponašanja

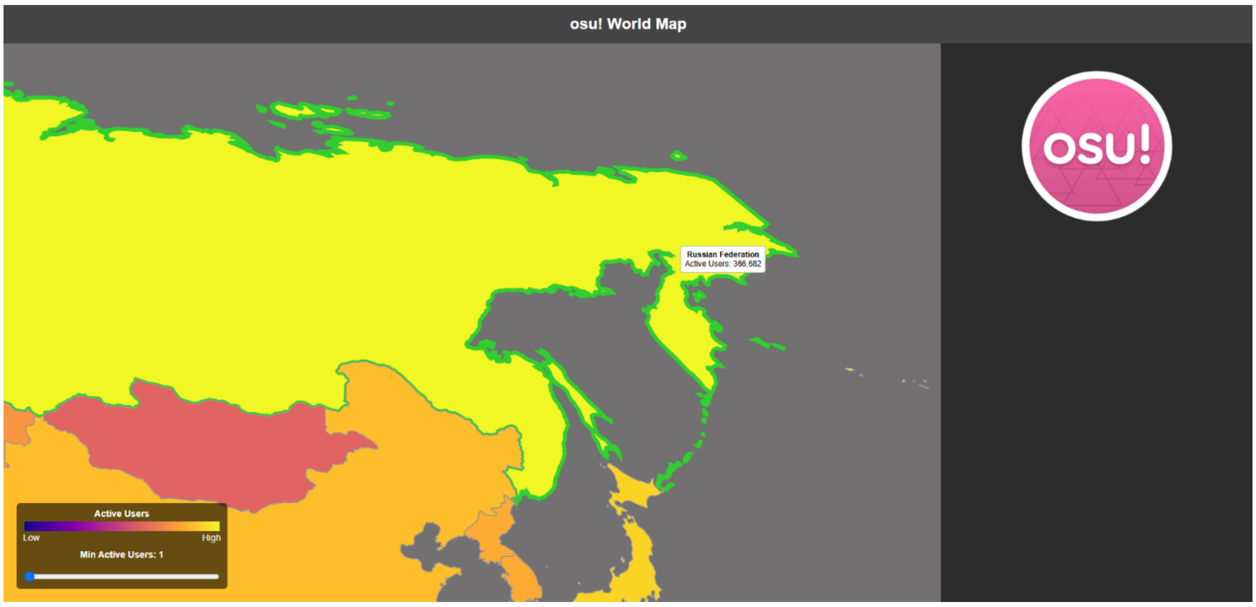
```
[const svg = d3.select("svg")
  .attr("width", width)
  .attr("height", height);


const g = svg.append("g");


const projection = d3.geoNaturalEarth1()
  .scale(width / 1.3 / Math.PI)
  .translate([width / 2, height / 2]);


const path = d3.geoPath(projection);


const zoom = d3.zoom()
  .scaleExtent([1, 8])
  .translateExtent([[-width * 0.5, -height * 0.5], [width * 1.5, height * 1.5]])
  .on("zoom", (event) => {
    g.attr("transform", event.transform);
  });

svg.call(zoom);

const countries = topojson.feature(worldData, worldData.objects.countries).features;

g.selectAll("path")
  .data(countries)
  .join("path")
  .attr("class", "country")
  .attr("d", path)
  .attr("fill", ...) // coloring logic
  .attr("stroke", "#999")
  .attr("stroke-width", 0.5)
```



4. KV4 - Izrada konačne vizualizacije podataka

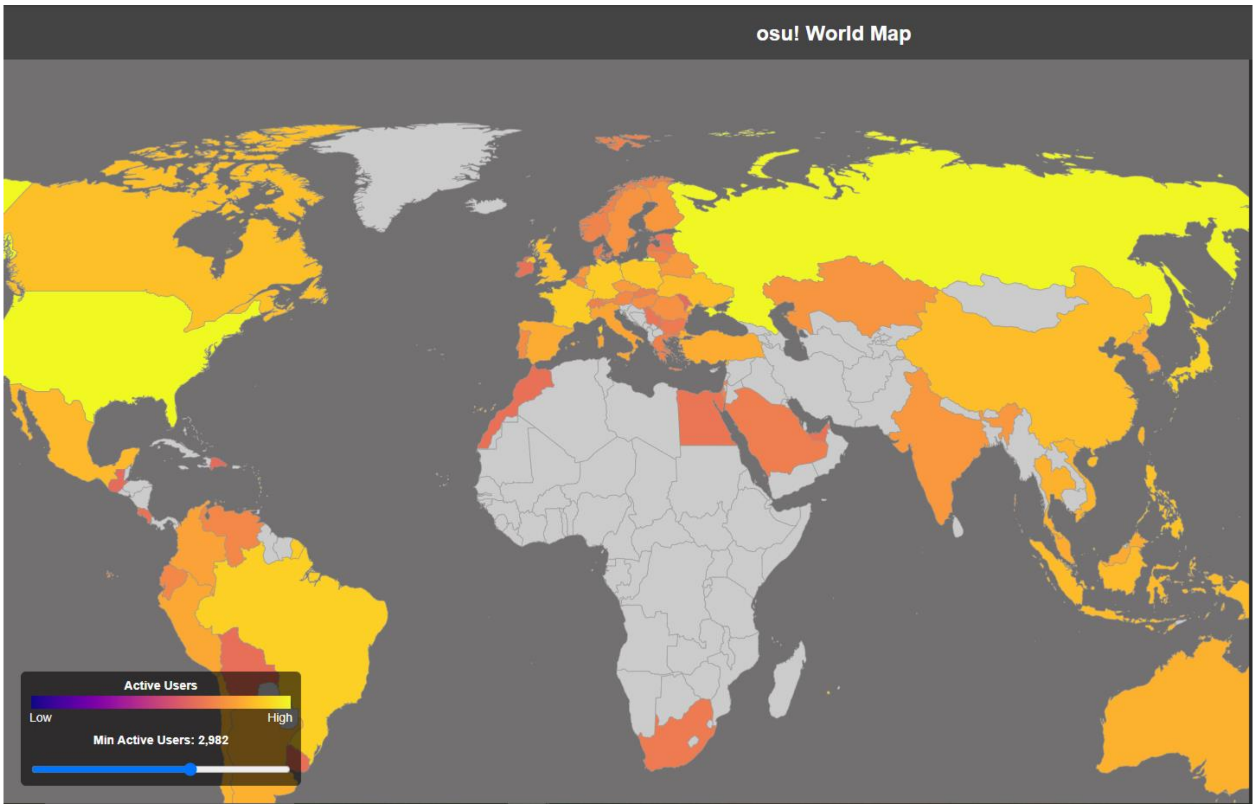
4.1. Implementacija osnovnih funkcionalnosti

4.2. Implementacija osnovnog ponašanja



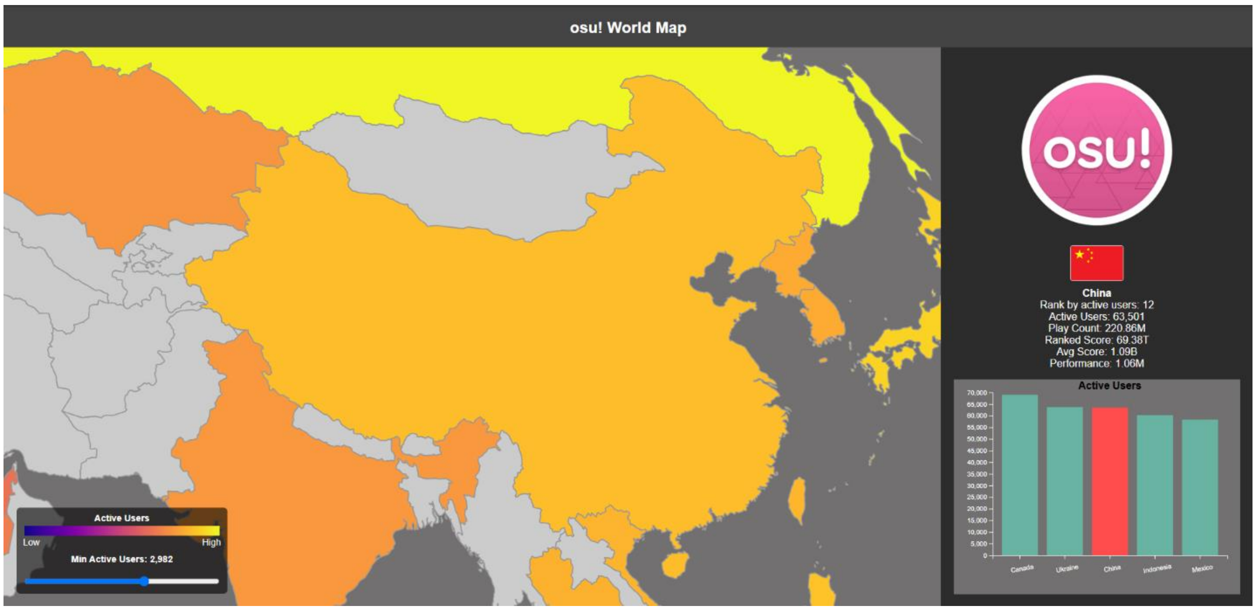
4.3. Implementacija naprednih funkcionalnosti

4.4. Implementacija naprednog ponašanja



Filtriranje user-a

```
<input type="range" id="minUsersRange" min="0" max="100" step="1" value="0" style="width: 300px;">
function sliderToUsers(sliderValue) {
  const logMin = Math.log10(minUsers);
  const logMax = Math.log10(maxUsers);
  const logValue = logMin + (logMax - logMin) * (sliderValue / 100);
  return Math.round(Math.pow(10, logValue));
}
let minUsersFilter = sliderToUsers(+slider.property("value"));
updateSliderDisplay(minUsersFilter);

function updateSliderDisplay(value) {
  d3.select("#minUsersValue").text(value.toLocaleString());
}
function updateMap() {
  g.selectAll("path.country")
    .attr("fill", d => {
      const countryInfo = countryById[+d.id];
      if (!countryInfo) return "#ccc";
      const users = countryData[countryInfo.name]?.activeUsers;
      if (!users || users < minUsersFilter) return "#ccc";
      return colorScale(users);
    });
}
slider.on("input", function() {
  const rawValue = +this.value;
  minUsersFilter = sliderToUsers(rawValue);
  updateSliderDisplay(minUsersFilter);
  updateMap();
});
```



China
Rank by active users: 12
Active Users: 63,501
Play Count: 220.86M
Ranked Score: 69.38T
Avg Score: 1.09B
Performance: 1.06M

Detaljniji podaci o izabranoj državi

```
.on("click", (event, d) => {
  event.stopPropagation();

  const countryInfo = countryById[+d.id];
  if (!countryInfo) return;
  const countryName = countryInfo.name;
  const alpha2 = countryInfo.alpha2.toLowerCase();  // for flag URL
  const data = countryData[countryName];

  // Zoom to country bounds
  const [[x0, y0], [x1, y1]] = path.bounds(d);
  svg.transition().duration(750).call(
    zoom.transform,
    d3.zoomIdentity
      .translate(width / 2, height / 2)
      .scale(Math.min(8, 0.9 / Math.max((x1 - x0) / width, (y1 - y0) / height)))
      .translate(-(x0 + x1) / 2, -(y0 + y1) / 2)
  );

  // Build flag image URL (flagcdn.com)
  const flagUrl = `https://flagcdn.com/w80/${alpha2}.png`;

  // Show country info with flag and stats (or "No data")
  if (data) {
    d3.select("#country-info").html(`
      <img src="${flagUrl}" alt="${countryName} flag" style="width:80px; height:auto; display:block;
margin: 0 auto 10px auto; border:1px solid #ccc; border-radius:4px;">
      <strong>${countryName}</strong><br/>
      Rank by active users: ${data.rank}<br/>
      Active Users: ${data.activeUsers.toLocaleString()}<br/>
      Total Score: ${data.totalScore}<br/>
      Ranked Score: ${data.rankedScore}<br/>
      Total Play Count: ${data.totalPlayCount}<br/>
      Performance: ${data.performance}`);
  } else {
    d3.select("#country-info").html(`
      <img src="${flagUrl}" alt="${countryName} flag" style="width:80px; height:auto; display:block;
margin: 0 auto 10px auto; border:1px solid #ccc; border-radius:4px;">
      <strong>${countryName}</strong><br/>No data available`);
    return;
  }

  // --- Bar chart for neighboring countries in rank ---

  const centerIndex = data.rank - 1;
  const start = Math.max(centerIndex - 2, 0);
  const end = start + 5;
  const neighbors = csvData.slice(start, end);

  const chartWidth = 360;
  const chartHeight = 250;
  const margin = { top: 20, right: 20, bottom: 60, left: 60 };

  // Append new SVG inside #country-info (this could create multiple charts on repeated clicks)
  const chartSvg = d3.select("#country-info")
    .append("div")
    .attr("id", "bar-chart")
    .append("svg")
    .attr("width", chartWidth + margin.left + margin.right)
    .attr("height", chartHeight + margin.top + margin.bottom)
    .append("g")
    .attr("transform", `translate(${margin.left},${margin.top})`);

  // Chart title
  chartSvg.append("text")
    .attr("x", chartWidth / 2)
    .attr("y", -5)
    .attr("text-anchor", "middle")
    .style("font-size", "16px")
    .style("font-weight", "bold")
    .text("Active Users");

  // X scale: countries names of neighbors
  const x = d3.scaleBand()
    .domain(neighbors.map(d => d.Country))
    .range([0, chartWidth])
    .padding(0.2);

  // Y scale: active users
  const y = d3.scaleLinear()
    .domain([0, d3.max(neighbors, d => parseActiveUsers(d["Active Users"]))])
    .nice()
    .range([chartHeight, 0]);

  // X axis
  chartSvg.append("g")
    .attr("transform", `translate(0,${chartHeight})`)
    .call(d3.axisBottom(x))
    .selectAll("text")
    .attr("transform", "rotate(-10)")
    .style("text-anchor", "middle")
    .attr("dy", "1.5em");

  // Y axis
  chartSvg.append("g")
    .call(d3.axisLeft(y));

  // Bars
  chartSvg.selectAll(".bar")
    .data(neighbors)
    .join("rect")
    .attr("class", "bar")
    .attr("x", d => x(d.Country))
    .attr("width", x.bandwidth())
    .attr("y", chartHeight)
    .attr("height", 0)
    .attr("fill", d => d.Country.trim() === countryName ? "#ff4d4d" : "#69b3a2")  // highlight clicked country
    .transition()
    .duration(800)
```

```
    .attr("y", d => y(parseActiveUsers(d["Active Users"])))
    .attr("height", d => chartHeight - y(parseActiveUsers(d["Active Users"])));
});
```

5. KV5 - Dovršetak projektnog zadatka i pisanje dokumentacije

5.1. Eventualne preinake i dorade rješenja - u dogovoru s nastavnikom

---

5.2. Izrada dokumenta - projektne dokumentacije

osu! world map je web vizualizacija temeljena na podacima koja prikazuje globalnu distribuciju aktivnih osu! igrača po državama. Boja mape prikazuje active user-s, te postoje interaktivne značajke poput zumiranja, pomicanja, legenda, filtriranje i statistike specifične za države.

Hierarhija:

country_ids.json

index.html

osu.py

osu_country_rankings.csv

osu_logo.png

style.css

Tehnologije:

HTML

CSS

JavaScript

D3.js

TopoJSON

World Atlas

SVG

Canvas API

DOM Events

JSON

CSV

ChatGPT

GitHub

GitHub Pages

FlagCDN

Upute:

Pokrenuti preko linka: https://paftdunk22.github.io/viz_pod_project/

Prikazuje se svjetska mapa po kojoj se može kretati mišem te zumirati

Hover na državu prikazuje osnovne podatke

Klikom na države se zumira na tu državu te ispisuju napredni podaci o njoj na desnom sidebar

Dolje lijevo legenda prikazuje bojanje mape te omogućava korisniku filtriranje broja igrača

Literatura

https://d3js.org/

https://github.com/d3/d3-geo

https://github.com/topojson/topojson

https://github.com/d3/d3-zoom

https://flagcdn.com/

https://chatgpt.com/

https://docs.python-requests.org/en/latest/

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

https://pandas.pydata.org/docs/

https://docs.python.org/3/library/csv.html

https://osu.ppy.sh/rankings/osu/country

https://moodle.srce.hr/2024-2025/pluginfile.php/11189445/mod_resource/content/0/Vizualizacija%20podataka%20LV%20priru%C4%8Dnik.pdf

Prilog I

Poveznica na git repozitorij projekta:
https://github.com/PaftDunk22/viz_pod_project

https://paftdunk22.github.io/viz_pod_project/

Programski kod:

Index.html:
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>OSU! World Map</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <script src="https://unpkg.com/topojson@3"></script>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>

<header>osu! World Map</header>

<div class="content">
  <svg></svg>
  <div class="plasma-legend">
    <div class="legend-title">Active Users</div>
    <canvas id="plasmaCanvas" width="300" height="15"></canvas>
    <div class="legend-labels">
      <span>Low</span>
      <span>High</span>
    </div>

    <div class="filter-active-users">
      <label for="minUsersRange">Min Active Users: <span id="minUsersValue">1</span></label><br>
      <input type="range" id="minUsersRange" min="0" max="100" step="1" value="0" style="width:
300px;">
    </div>

  </div>

  <div class="sidebar">
    <img src="osu_logo.png" alt="osu! logo">
    <div id="country-info" style="margin-top: 20px; font-size: 16px; text-align: center;"></div>
    <div id="bar-chart" style="width: 100%; height: 300px; margin-top: 20px;"></div>
  </div>

  <div class="tooltip"></div>

<script>
```

```
const width = window.innerWidth * 0.8;
const height = window.innerHeight - document.querySelector('header').offsetHeight;

const svg = d3.select("svg")
  .attr("width", width)
  .attr("height", height);

const g = svg.append("g");

const projection = d3.geoNaturalEarth1()
  .scale(width / 1.3 / Math.PI)
  .translate([width / 2, height / 2]);

const path = d3.geoPath(projection);
const tooltip = d3.select(".tooltip");

const zoom = d3.zoom()
  .scaleExtent([1, 8])
  .translateExtent([[-width * 0.5, -height * 0.5], [width * 1.5, height * 1.5]])
  .on("zoom", (event) => {
    g.attr("transform", event.transform);
  });

svg.call(zoom);

let countryById = {};

function parseActiveUsers(str) {
  if (!str) return 0;
  return +str.replace(/,/g, '');
}

Promise.all([
  d3.csv("osu_country_rankings.csv"),
  d3.json("https://cdn.jsdelivr.net/npm/world-atlas@2/countries-50m.json"),
  d3.json("country_ids.json")
]).then(([csvData, worldData, countryMap]) => {
  countryById = countryMap;

  const countryData = {};
  csvData.forEach((d, i) => {
    countryData[d.Country.trim()] = {
      rank: i + 1,
      activeUsers: parseActiveUsers(d["Active Users"]),
      totalScore: d["Total Score"],
      rankedScore: d["Ranked Score"],
      totalPlayCount: d["Total Play Count"],
      performance: d["Performance"]
    };
  });
```

```
const minUsers = 1;
const userValues = Object.values(countryData)
  .map(d => d.activeUsers)
  .filter(d => d > 0);
const maxUsers = d3.max(userValues);

const colorScale = d3.scaleSequentialLog()
  .domain([minUsers, maxUsers])
  .interpolator(d3.interpolatePlasma);

const slider = d3.select("#minUsersRange");

function sliderToUsers(sliderValue) {
  const logMin = Math.log10(minUsers);
  const logMax = Math.log10(maxUsers);
  const logValue = logMin + (logMax - logMin) * (sliderValue / 100);
  return Math.round(Math.pow(10, logValue));
}

function updateSliderDisplay(value) {
  d3.select("#minUsersValue").text(value.toLocaleString());
}

let minUsersFilter = sliderToUsers(+slider.property("value"));
updateSliderDisplay(minUsersFilter);

const countries = topojson.feature(worldData, worldData.objects.countries).features;

function updateMap() {
  g.selectAll("path.country")
    .attr("fill", d => {
      const countryInfo = countryById[+d.id];
      if (!countryInfo) return "#ccc";
      const users = countryData[countryInfo.name]?.activeUsers;
      if (!users || users < minUsersFilter) return "#ccc";
      return colorScale(users);
    });
}

g.selectAll("path")
  .data(countries)
  .join("path")
  .attr("class", "country")
  .attr("d", path)
  .attr("fill", d => {
    const countryInfo = countryById[+d.id];
    if (!countryInfo) return "#ccc";
    const users = countryData[countryInfo.name]?.activeUsers;
    if (!users || users < minUsersFilter) return "#ccc";
    return colorScale(users);
  })
```

```
    .attr("stroke", "#999")
    .attr("stroke-width", 0.5)
    .on("mouseover", (event, d) => {
      const countryInfo = countryById[+d.id];
      const countryName = countryInfo ? countryInfo.name : "Unknown";
      const data = countryData[countryName];
      const users = data?.activeUsers ?? "No data";

      d3.select(event.currentTarget)
        .attr("stroke", "limegreen")
        .attr("stroke-width", 2);

      tooltip.style("opacity", 1)
        .html(`<strong>${countryName}</strong><br/>Active Users: ${typeof users === "number" ?
users.toLocaleString() : users}`)
        .style("left", (event.pageX + 10) + "px")
        .style("top", (event.pageY - 28) + "px");
    })
    .on("mouseout", (event, d) => {
      d3.select(event.currentTarget)
        .attr("stroke", "#999")
        .attr("stroke-width", 0.5);

      tooltip.style("opacity", 0);
    })
    .on("click", (event, d) => {
      event.stopPropagation();

      const countryInfo = countryById[+d.id];
      if (!countryInfo) return;
      const countryName = countryInfo.name;
      const alpha2 = countryInfo.alpha2.toLowerCase();
      const data = countryData[countryName];

      const [[x0, y0], [x1, y1]] = path.bounds(d);
      svg.transition().duration(750).call(
        zoom.transform,
        d3.zoomIdentity
          .translate(width / 2, height / 2)
          .scale(Math.min(8, 0.9 / Math.max((x1 - x0) / width, (y1 - y0) / height)))
          .translate(-(x0 + x1) / 2, -(y0 + y1) / 2)
      );

      const flagUrl = `https://flagcdn.com/w80/${alpha2}.png`;

      if (data) {
        d3.select("#country-info").html(`
          <img src="${flagUrl}" alt="${countryName} flag" style="width:80px; height:auto; display:block;
margin: 0 auto 10px auto; border:1px solid #ccc; border-radius:4px;">
          <strong>${countryName}</strong><br/>
          Rank by active users: ${data.rank}<br/>
          Active Users: ${data.activeUsers.toLocaleString()}<br/>
          Total Score: ${data.totalScore}<br/>
          Ranked Score: ${data.rankedScore}<br/>
          Total Play Count: ${data.totalPlayCount}<br/>
          Performance: ${data.performance}`);

      } else {
        d3.select("#country-info").html(`
          <img src="${flagUrl}" alt="${countryName} flag" style="width:80px; height:auto; display:block;
margin: 0 auto 10px auto; border:1px solid #ccc; border-radius:4px;">
          <strong>${countryName}</strong><br/>No data available`);
        return;
      }

      const centerIndex = data.rank - 1;
      const start = Math.max(centerIndex - 2, 0);
      const end = start + 5;
      const neighbors = csvData.slice(start, end);

      const chartWidth = 360;
      const chartHeight = 250;
      const margin = { top: 20, right: 20, bottom: 60, left: 60 };

      const chartSvg = d3.select("#country-info")
        .append("div")
        .attr("id", "bar-chart")
        .append("svg")
        .attr("width", chartWidth + margin.left + margin.right)
        .attr("height", chartHeight + margin.top + margin.bottom)
        .append("g")
        .attr("transform", `translate(${margin.left},${margin.top})`);

      chartSvg.append("text")
        .attr("x", chartWidth / 2)
        .attr("y", -5)
        .attr("text-anchor", "middle")
        .style("font-size", "16px")
        .style("font-weight", "bold")
        .text("Active Users");

      const x = d3.scaleBand()
        .domain(neighbors.map(d => d.Country))
        .range([0, chartWidth])
        .padding(0.2);

      const y = d3.scaleLinear()
        .domain([0, d3.max(neighbors, d => parseActiveUsers(d["Active Users"]))])
        .nice()
        .range([chartHeight, 0]);

      chartSvg.append("g")
```

```
        .attr("transform", `translate(0,${chartHeight})`)
        .call(d3.axisBottom(x))
        .selectAll("text")
        .attr("transform", "rotate(-10)")
        .style("text-anchor", "middle")
        .attr("dy", "1.5em");

      chartSvg.append("g")
        .call(d3.axisLeft(y));

      chartSvg.selectAll(".bar")
        .data(neighbors)
        .join("rect")
        .attr("class", "bar")
        .attr("x", d => x(d.Country))
        .attr("width", x.bandwidth())
        .attr("y", chartHeight)
        .attr("height", 0)
        .attr("fill", d => d.Country.trim() === countryName ? "#ff4d4d" : "#69b3a2")
        .transition()
        .duration(800)
        .attr("y", d => y(parseActiveUsers(d["Active Users"])))
        .attr("height", d => chartHeight - y(parseActiveUsers(d["Active Users"])));
    });

  svg.on("click", () => {
    svg.transition().duration(750).call(
      zoom.transform,
      d3.zoomIdentity
    );
    d3.select("#country-info").html("");
  });

  slider.on("input", function() {
    const rawValue = +this.value;
    minUsersFilter = sliderToUsers(rawValue);
    updateSliderDisplay(minUsersFilter);
    updateMap();
  });

});

function drawPlasmaLegend() {
  const canvas = document.getElementById("plasmaCanvas");
  const ctx = canvas.getContext("2d");
  const width = canvas.width;
  const height = canvas.height;

  const gradient = ctx.createLinearGradient(0, 0, width, 0);
  for (let i = 0; i <= 100; i++) {
    const t = i / 100;
```

```
      gradient.addColorStop(t, d3.interpolatePlasma(t));
    }

    ctx.fillStyle = gradient;
    ctx.fillRect(0, 0, width, height);
  }

  drawPlasmaLegend();

</script>
</body>
</html>
```

osu.py:

```python
import requests
from bs4 import BeautifulSoup
import csv
import time
import pandas

def scrape_page(page_number):
    url = f"https://osu.ppy.sh/rankings/osu/country?page={page_number}#scores"
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    rows = soup.find_all('tr', class_='ranking-page-table__row')

    country_data = []

    for row in rows:
        try:
            country_name = row.find_all('td', class_='ranking-page-table__column')[1].text.strip()
            active_users = row.find_all('td', class_='ranking-page-table__column')[2].text.strip()
            total_score = row.find_all('td', class_='ranking-page-table__column')[3].text.strip()
            ranked_score = row.find_all('td', class_='ranking-page-table__column')[4].text.strip()
            total_play_count = row.find_all('td', class_='ranking-page-table__column')[5].text.strip()
            performance = row.find_all('td', class_='ranking-page-table__column')[6].text.strip()

            country_data.append([country_name, active_users, total_score, ranked_score,
total_play_count, performance])
        except Exception as e:
            print(f"Error scraping row: {e}")

    return country_data

csv_filename = "osu_country_rankings.csv"
csv_file = open(csv_filename, mode='w', newline='', encoding='utf-8')
csv_writer = csv.writer(csv_file)

csv_writer.writerow(['Country', 'Active Users', 'Total Score', 'Ranked Score', 'Total Play Count',
'Performance'])
```

```
for page in range(1, 6):
    print(f"Scraping page {page}...")
    country_data = scrape_page(page)

    if country_data:
        csv_writer.writerows(country_data)
    else:
        print(f"No data found on page {page}.")

    time.sleep(2)


csv_file.close()

df = pandas.read_csv(csv_filename)

df["Active Users"] = df["Active Users"].str.replace(",", "").astype(int)

df = df.sort_values("Active Users", ascending=False)

df.to_csv(csv_filename, index=False)

print("Data has been saved to osu_country_rankings.csv")
```