LAMPIRAN

Lampiran 1

Surat Keterangan Bebas Plagiat

UNIVERSITAS DARMA PERSADA
UPT PERPUSTAKAAN
Gedung Rektorat Lantai 3,
Jl.Taman Malaka Selatan, Pondok Kelapa – Jakarta Timur 13450

SURAT KETERANGAN
HASIL PENGECEKAN TURNITIN

UPT Perpustakaan Universitas Darma Persada menerangkan telah selesai melakukan pemeriksaan duplikasi/*similarity* menggunakan perangkat lunak Turnitin terhadap hasil karya sebagai berikut:

Judul	: Perancangan Sistem Aplikasi Penjadwalan Sidang Skripsi Berbasis Web Di Platform Mobile Menggunakan Algoritma Genetika
Penulis	: Yudha Julio Pratama
NIM	: **2021230026**
Tgl pemeriksaan	: 28 Juli 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **8%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 28 Juli 2025

Ka.UPT Perpustakaan Unsada

Yus Rusmiyati, SS., MM

Batas maksimal similarity 30% untuk Fakultas Sastra dan Ekonomi

Batas maksimal similarity 25% untuk Fakultas Teknik, Kelautan dan Pasca Sarjana

Lampiran 2

Source Code GA

```
// Fungsi Utama GA
const generateGA = async (datasetOriginal) => {
  let dataset = [...datasetOriginal];

  if (selectedCategory === "Sempro") {
    dataset = dataset.filter(m => m.statusSempro === "Masih
Disidangkan");
    if (dataset.length === 0) {
      alert("Tidak ada mahasiswa Sempro dengan status 'Masih
Disidangkan'");
      return;
    }
  }
  if (listDosen.length < 4) {
    alert("Minimal butuh 4 dosen untuk pembimbing + penguji.");
    return;
  }
  const tanggalSidang = getTanggalSidang(dataset.length);
  const jumlahPopulasi = 20;
  const totalGenerasi = 10;
  let populasi = [], log = [], chartLog = [], steps = [];
  const getPengujiSempro = (namaMahasiswa) => {
    const mhs = dataSempro.find(d => d.nama === namaMahasiswa);
    return {
      penguji1: mhs?.penguji1 || listDosen[Math.floor(Math.random() *
listDosen.length)],
      penguji2: mhs?.penguji2 || listDosen[Math.floor(Math.random() *
listDosen.length)],
    };
  };
  const createRandomSolution = () => dataset.map((mhs) => {
    const waktu = waktuSidang[Math.floor(Math.random() *
waktuSidang.length)];
    const tanggal = tanggalSidang[Math.floor(Math.random() *
tanggalSidang.length)];
    const ruang = ruangan[Math.floor(Math.random() * ruangan.length)];
```

```
    const pembimbing = mhs.dosen || listDosen[Math.floor(Math.random()
* listDosen.length)];
    const judul = mhs.judul || "-";
    if (selectedCategory === "Sempro") {
      let penguji1 = pembimbing, penguji2 = pembimbing;
      while (penguji1 === pembimbing) penguji1 =
listDosen[Math.floor(Math.random() * listDosen.length)];
      while (penguji2 === pembimbing || penguji2 === penguji1) penguji2
= listDosen[Math.floor(Math.random() * listDosen.length)];
      return { mahasiswaId: mhs.id, namaMahasiswa: mhs.nama, tanggal,
jam: waktu, ruangan: ruang, pembimbing, penguji1, penguji2, judul };
    }
    // if (selectedCategory === "SeminarIsi" || selectedCategory ===
"Skripsi") {
    //   const { penguji1, penguji2 } = getPengujiSempro(mhs.nama);
    //   return {
    //     mahasiswaId: mhs.id, namaMahasiswa: mhs.nama,
    //     tanggal, jam: waktu, ruangan: ruang,
    //     pembimbing,
    //     penguji1: pembimbing,
    //     penguji2,
    //     penguji3: listDosen[Math.floor(Math.random() *
listDosen.length)]
    //   };
    // }
     if (selectedCategory === "SeminarIsi" || selectedCategory ===
"Skripsi") {
      // Butuh 3 penguji, tidak boleh sama satu sama lain & pembimbing
      let penguji1 = pembimbing;
      let penguji2 = pembimbing;
      let penguji3 = pembimbing;
      while (penguji1 === pembimbing) {
        penguji1 = listDosen[Math.floor(Math.random() *
listDosen.length)];
      }
      while (penguji2 === pembimbing || penguji2 === penguji1) {
        penguji2 = listDosen[Math.floor(Math.random() *
listDosen.length)];
      }
```

```
      while (
        penguji3 === pembimbing ||
        penguji3 === penguji1 ||
        penguji3 === penguji2
      ) {
        penguji3 = listDosen[Math.floor(Math.random() *
listDosen.length)];
      }
      return {
        mahasiswaId: mhs.id,
        namaMahasiswa: mhs.nama,
        tanggal,
        jam: waktu,
        ruangan: ruang,
        pembimbing,
        judul,
        penguji1,
        penguji2,
        penguji3,
      };
    }
    return null; // fallback
  });
  // for (let i = 0; i < jumlahPopulasi; i++) {
  //   let solusi = createRandomSolution();
  //   solusi = resolveBentrokDenganPindahHari(solusi, tanggalSidang,
10, 15, 4);
  //   const fitness = hitungFitness(solusi);
  //   populasi.push({ solusi, fitness });
  // }
  for (let i = 0; i < jumlahPopulasi; i++) {
    let solusi = createRandomSolution();
    solusi = resolveBentrokDenganPindahHari(solusi, tanggalSidang, 10,
15, 4);
    const hasil = hitungFitness(solusi);
    populasi.push({ solusi, fitness: hasil.score, detailFitness:
hasil.detail });
  }
  for (let g = 0; g < totalGenerasi; g++) {
```

```
    populasi.sort((a, b) => b.fitness - a.fitness);
    const best = populasi[0];
    log.push(Generasi ${g + 1}: fitness terbaik ${best.fitness});
    chartLog.push({ generasi: Gen-${g + 1}, fitness: best.fitness });
    await addDoc(collection(db, "riwayat_GA"), {
      waktu: new Date().toISOString(),
      kategori: selectedCategory,
      generasi: g + 1,
      fitness: best.fitness,
      jadwal: best.solusi,
      timestamp: new Date(),  // <= wajib
    });
    steps.push(' Seleksi Gen-${g + 1}: ${best.fitness});
    const selectParent = () => {
      const kandidat = [
        populasi[Math.floor(Math.random() * populasi.length)],
        populasi[Math.floor(Math.random() * populasi.length)]
      ];
      return kandidat.sort((a, b) => b.fitness - a.fitness)[0].solusi;
    };
    const parentA = selectParent();
    const parentB = selectParent();
    const crossoverPoint = Math.floor(parentA.length / 2);
    let child = [...parentA.slice(0, crossoverPoint),
...parentB.slice(crossoverPoint)].map(gene => ({ ...gene }));
    steps.push( Crossover Gen-${g + 1}: titik potong
${crossoverPoint});
    for (let i = 0; i < child.length; i++) {
      if (Math.random() < 0.1) {
        const tipeMutasi = Math.floor(Math.random() * 5);
        if (tipeMutasi === 0) child[i].jam =
waktuSidang[Math.floor(Math.random() * waktuSidang.length)];
        if (tipeMutasi === 1) child[i].ruangan =
ruangan[Math.floor(Math.random() * ruangan.length)];
        if (tipeMutasi === 2) child[i].pembimbing =
listDosen[Math.floor(Math.random() * listDosen.length)];
        if (tipeMutasi === 3) child[i].penguji1 =
listDosen[Math.floor(Math.random() * listDosen.length)];
        if (tipeMutasi === 4 && child[i].penguji2) child[i].penguji2 =
```

```
listDosen[Math.floor(Math.random() * listDosen.length)];
      }
    }
    steps.push(`Mutasi Gen-${g + 1}`);
    child = resolveBentrokDenganPindahHari(child, tanggalSidang, 10,
15, 4);
    // const fitnessAnak = hitungFitness(child);
    // populasi.pop();
    // populasi.push({ solusi: child, fitness: fitnessAnak });
    const hasilAnak = hitungFitness(child);
    populasi.pop();
    populasi.push({ solusi: child, fitness: hasilAnak.score,
detailFitness: hasilAnak.detail });
  }
  populasi.sort((a, b) => b.fitness - a.fitness);
  setJadwalTerbaik(populasi[0].solusi);
  setFitnessTerbaik(populasi[0].fitness);
  setDetailFitnessTerbaik(populasi[0].detailFitness); // ← tambahkan
ini
  setLogGenerasi(log);
  setChartData(chartLog);
  setProcessSteps(steps);
  const nilaiFitness = chartLog.map(d => d.fitness);
const bestFitness = Math.max(...nilaiFitness);
const worstFitness = Math.min(...nilaiFitness);
const averageFitness = (
  nilaiFitness.reduce((a, b) => a + b, 0) / nilaiFitness.length
).toFixed(2);
setStatistikFitness({
  best: bestFitness.toFixed(2),
  avg: averageFitness,
  worst: worstFitness.toFixed(2),
});
  const dosenCounter = {};
  populasi[0].solusi.forEach((entry) => {
    [entry.pembimbing, entry.penguji1, entry.penguji2, entry.penguji3]
      .filter(Boolean)
      .forEach(d => dosenCounter[d] = (dosenCounter[d] || 0) + 1);
  });
```

```
  setDosenLoad(dosenCounter);
};
<div className="bg-white rounded-2xl shadow-lg p-6 space-y-6 w-full
max-w-6xl mx-auto">
      <h2 className="text-2xl font-bold text-gray-800">Analisis
Algoritma Genetika</h2>
      <p className="text-sm text-gray-500">Perkembangan Nilai
Fitness</p>
      <div className="grid grid-cols-1 sm:grid-cols-3 gap-4">
        <div className="bg-blue-50 rounded-xl p-4 shadow-sm border
border-blue-100">
          <h3 className="text-sm text-gray-600">Best Fitness</h3>
          <p className="text-2xl font-semibold text-blue-
700">{best.toFixed(2)}</p>
        </div>
<div className="bg-purple-50 rounded-xl p-4 shadow-sm border border-
purple-100">
          <h3 className="text-sm text-gray-600">Average Fitness</h3>
<p className="text-2xl font-semibold text-purple-
700">{avg.toFixed(2)}</p>
        </div>
        <div className="bg-red-50 rounded-xl p-4 shadow-sm border
border-red-100">
          <h3 className="text-sm text-gray-600">Worst Fitness</h3>
          <p className="text-2xl font-semibold text-red-
700">{worst.toFixed(2)}</p>
        </div>
      </div>
      <div className="mt-4">
        <LineChart width={600} height={300} data={chartData}>
          <CartesianGrid stroke="#ccc" />
          <XAxis dataKey="generation" />
          <YAxis />
          <Tooltip />
          <Legend />
<Line type="monotone" dataKey="best" stroke="#3b82f6" name="Best" />
<Line type="monotone" dataKey="average" stroke="#8b5cf6" name="Average"
/> </div>
```