ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Програмні засоби моніторингу інформаційної системи

Кваліфікаційна робота

Виконав:
студент гр. КІУКІ-21-1
Колобаєв Н.М.

Керівник:
ас. каф. ЕОМ,
Климова І.М.

Мета роботи та завдання 2

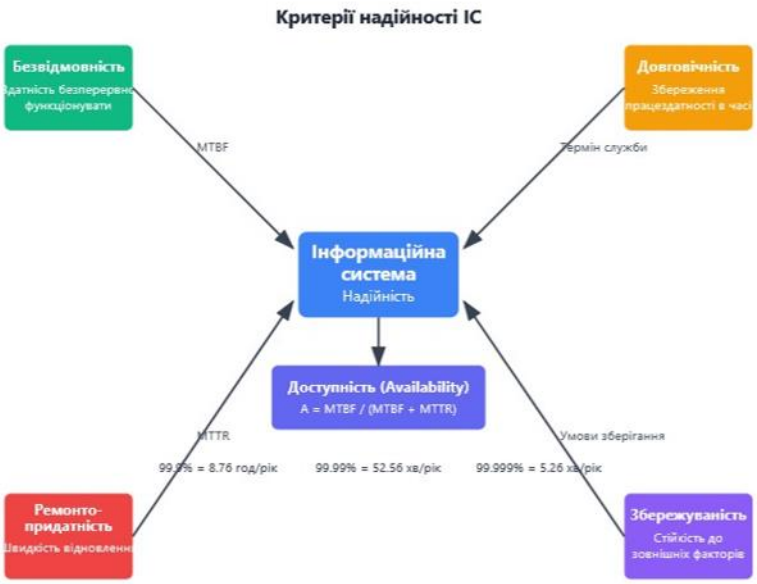
Метою кваліфікаційної роботи є аналіз сучасних підходів до моніторингу стану інформаційних систем, а також розробка програмних засобів моніторингу стану інформаційної системи.

Завдання:

- ❑ провести теоретичний аналіз концептуальних основ моніторингу інформаційних систем;
- ❑ Провести порівняльний аналіз архітектурних рішень;
- ❑ вибір засобів реалізації ПЗ;
- ❑ практична реалізація програмних засобів моніторингу стану інформаційної системи;
- ❑ аналіз результатів

Порівняльний аналіз методів машинного навчання 3

Критерії надійності ІС 4



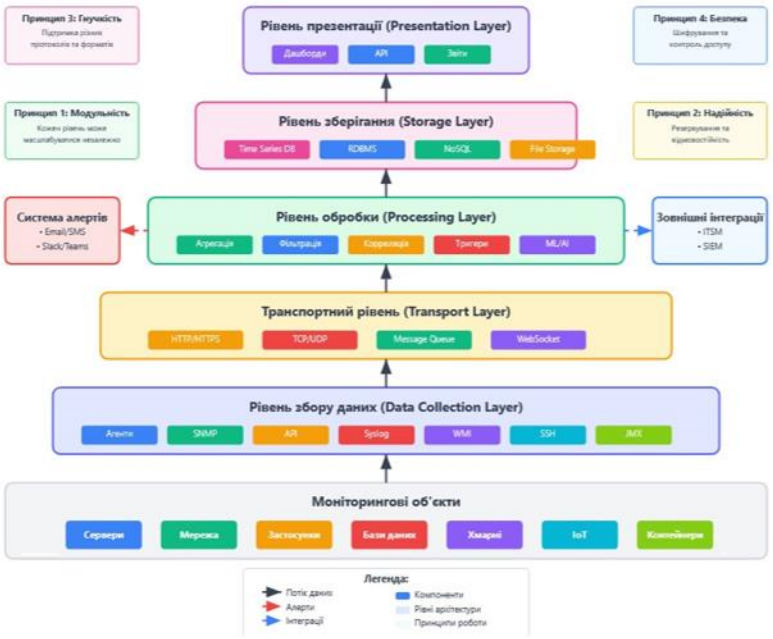
5

Типи та рівні моніторнгу

6

Порівняльний аналіз рішень

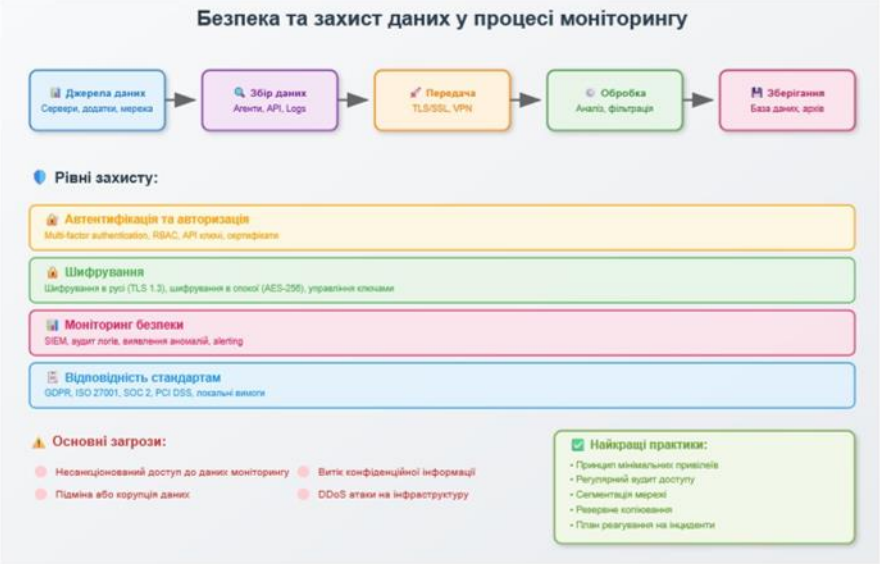
Архітектура систем моніторингу 7



Проблеми масштабування та продуктивності ІС 8

9

Безпека та захист даних



10

Архітектура ПЗ

Технічні характеристики:

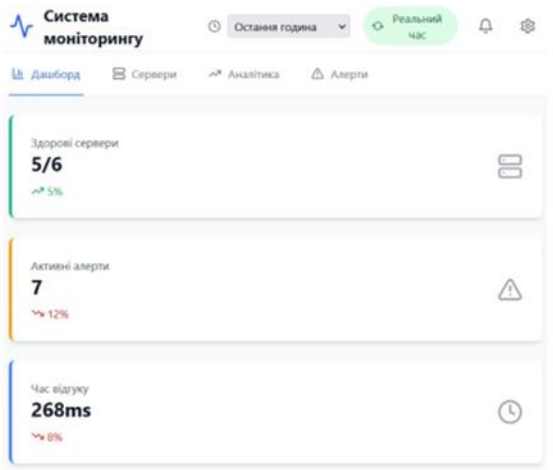
- Оновлення даних кожні 5 секунд (setInterval)
- Генерація даних: 6 серверів, 10 алертів, 5 сервісів
- Часові ряди: 60 точок для метрик (остання година)
- Інтерактивність: пошук, фільтрація, навігація табами

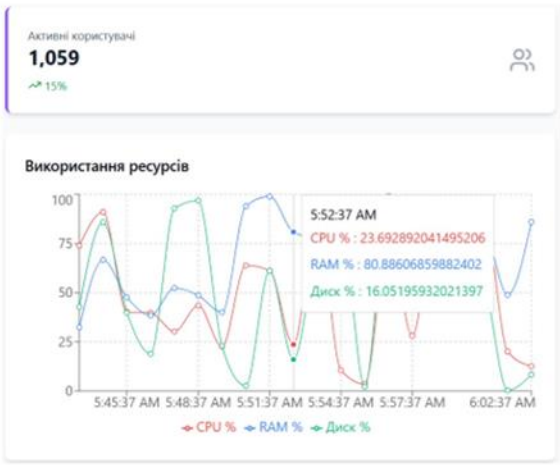
Реалізовані функції:

- 4 інтерактивні дашборди (Dashboard, Servers, Analytics, Alerts)
- 8 типів графіків (LineChart, AreaChart, BarChart, PieChart)
- Реальний час: автоматичне оновлення та Live індикатори
- Система алертів з класифікацією за важливістю

11

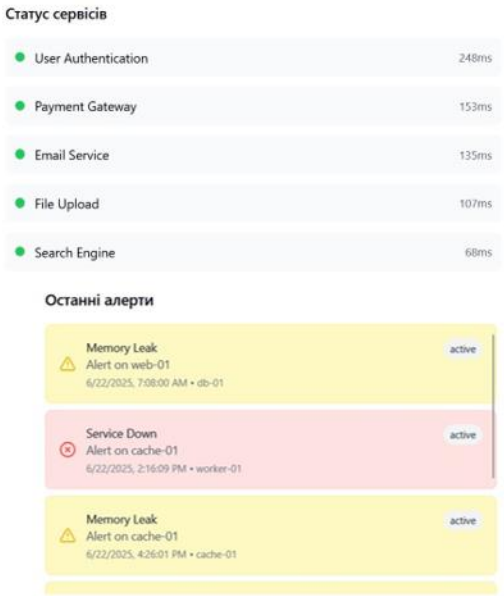
Результати роботи





12

Результати роботи



Висновки

13

У процесі виконання кваліфікаційної роботи було програмні засоби моніторингу стану інформаційної системи. У роботі досліджено сучасні підходи до моніторингу стану інформаційних систем та проаналізовано програмні засоби, що забезпечують ефективний контроль функціонування IT-інфраструктури. Проведено комплексний аналіз теоретичних основ моніторингу інформаційних систем, включаючи дослідження концептуальних засад, критеріїв надійності та доступності систем, а також класифікацію існуючих рішень.

Виконано порівняльний аналіз провідних програмних засобів моніторингу, таких як Zabbix, Nagios, Prometheus та Grafana, з детальним вивченням їх архітектурних особливостей, функціональних можливостей та принципів роботи. Особливу увагу приділено проблемам масштабування, продуктивності та забезпечення безпеки в контексті сучасних вимог до систем моніторингу.

ДОДАТОК Б

Програмний код

Б.1 Лістинг коду

```
npm install recharts lucide-react
npm install -D tailwindcss
// Запустили, тепер сам код
import React, { useState, useEffect, useRef } from 'react';
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip,
Legend, ResponsiveContainer, AreaChart, Area, BarChart, Bar,
PieChart, Pie, Cell } from 'recharts';
import { AlertTriangle, CheckCircle, XCircle, Activity, Server,
Database, Wifi, Cpu, HardDrive, MemoryStick, Clock, Users, Bell,
Settings, TrendingUp, TrendingDown, Eye, EyeOff, RefreshCw,
Download, Filter, Search, Calendar, BarChart3 } from 'lucide-
react';

const MonitoringSystem = () => {
  const [activeTab, setActiveTab] = useState('dashboard');
  const [isRealTime, setIsRealTime] = useState(true);
  const [selectedTimeRange, setSelectedTimeRange] =
useState('1h');
  const [searchTerm, setSearchTerm] = useState('');
  const [filterStatus, setFilterStatus] = useState('all');
  const [notifications, setNotifications] = useState([]);
  const [systemData, setSystemData] = useState({
    servers: [],
    metrics: [],
    alerts: [],
    services: [],
    networkData: [],
    performanceData: []
  });

  // Симуляція даних для демонстрації
  const generateRandomData = () => {
    const now = Date.now();
    const metrics = [];
    const servers = [];
    const alerts = [];
    const services = [];
    const networkData = [];
    const performanceData = [];

    // Генерація метрик для останньої години
    for (let i = 59; i >= 0; i--) {
```

```
      const timestamp = now - i * 60000;
      metrics.push({
        time: new Date(timestamp).toLocaleTimeString(),
        timestamp,
        cpu: Math.random() * 100,
        memory: Math.random() * 100,
        disk: Math.random() * 100,
        network: Math.random() * 1000,
        responseTime: Math.random() * 500 + 50,
        throughput: Math.random() * 10000 + 1000,
        errors: Math.floor(Math.random() * 10),
        activeUsers: Math.floor(Math.random() * 1000) + 100
      });
    }

    // Генерація даних серверів
    const serverNames = ['web-01', 'web-02', 'db-01', 'cache-
01', 'api-01', 'worker-01'];
    serverNames.forEach((name, index) => {
      const status = Math.random() > 0.8 ? 'warning' :
Math.random() > 0.9 ? 'critical' : 'healthy';
      servers.push({
        id: index + 1,
        name,
        status,
        cpu: Math.random() * 100,
        memory: Math.random() * 100,
        disk: Math.random() * 100,
        uptime: Math.floor(Math.random() * 365) + 1,
        location: ['Kyiv', 'Dnipro',
'Lviv'][Math.floor(Math.random() * 3)],
        type: ['Web Server', 'Database', 'Cache', 'API Server',
'Worker'][Math.floor(Math.random() * 5)]
      });
    });

    // Генерація алертів
    const alertTypes = ['High CPU Usage', 'Memory Leak', 'Disk
Space Low', 'Network Timeout', 'Service Down'];
    for (let i = 0; i < 10; i++) {
      const severity = ['critical', 'warning',
'info'][Math.floor(Math.random() * 3)];
      alerts.push({
        id: i + 1,
        type: alertTypes[Math.floor(Math.random() *
alertTypes.length)],
        message: `Alert on
${serverNames[Math.floor(Math.random() * serverNames.length)]}`,
        severity,
        timestamp: new Date(now - Math.random() *
86400000).toISOString(),
        status: Math.random() > 0.7 ? 'resolved' : 'active',
        server: serverNames[Math.floor(Math.random() *
```

```
serverNames.length)]
      });
    }

    // Генерація даних сервісів
    const serviceNames = ['User Authentication', 'Payment
Gateway', 'Email Service', 'File Upload', 'Search Engine'];
    serviceNames.forEach((name, index) => {
      const status = Math.random() > 0.9 ? 'down' :
Math.random() > 0.8 ? 'degraded' : 'operational';
      services.push({
        id: index + 1,
        name,
        status,
        responseTime: Math.random() * 200 + 50,
        uptime: Math.random() * 100,
        requests: Math.floor(Math.random() * 10000) + 1000,
        errors: Math.floor(Math.random() * 50)
      });
    });

    // Генерація мережевих даних
    for (let i = 23; i >= 0; i--) {
      const timestamp = now - i * 3600000;
      networkData.push({
        time: new Date(timestamp).getHours() + ':00',
        timestamp,
        inbound: Math.random() * 1000 + 100,
        outbound: Math.random() * 800 + 50,
        latency: Math.random() * 100 + 10,
        packetLoss: Math.random() * 5
      });
    }

    // Генерація даних продуктивності
    const performanceMetrics = ['Database Queries', 'Cache
Hits', 'API Calls', 'File Operations'];
    performanceMetrics.forEach(metric => {
      for (let i = 11; i >= 0; i--) {
        const timestamp = now - i * 300000; // кожні 5 хвилин
        performanceData.push({
          metric,
          time: new Date(timestamp).toLocaleTimeString(),
          timestamp,
          value: Math.random() * 1000 + 100,
          success: Math.random() * 100,
          failed: Math.random() * 10
        });
      }
    });

    setSystemData({
      servers,
```

```
      metrics,
      alerts,
      services,
      networkData,
      performanceData
    });
  };

  // Ініціалізація та оновлення даних
  useEffect(() => {
    generateRandomData();

    if (isRealTime) {
      const interval = setInterval(generateRandomData, 5000);
      return () => clearInterval(interval);
    }
  }, [isRealTime]);

  // Компонент статистичної карточки
  const StatCard = ({ title, value, icon: Icon, trend, color =
'blue' }) => (
    <div className="bg-white rounded-lg shadow-md p-6 border-l-
4" style={{ borderLeftColor: color }}>
      <div className="flex items-center justify-between">
        <div>
          <p className="text-sm font-medium text-gray-
600">{title}</p>
          <p className="text-2xl font-bold text-gray-
900">{value}</p>
          {trend && (
            <div className={`flex items-center mt-2 ${trend > 0
? 'text-green-600' : 'text-red-600'}`}>
              {trend > 0 ? <TrendingUp size={16} /> :
<TrendingDown size={16} />}
              <span className="ml-1 text-
sm">{Math.abs(trend)}%</span>
            </div>
          )}
        </div>
        <Icon size={32} className="text-gray-400" />
      </div>
    </div>
  );

  // Компонент алерту
  const AlertItem = ({ alert }) => {
    const getAlertIcon = (severity) => {
      switch (severity) {
        case 'critical': return <XCircle className="text-red-
500" size={20} />;
        case 'warning': return <AlertTriangle className="text-
yellow-500" size={20} />;
        default: return <CheckCircle className="text-blue-500"
```

```
size={20} />;
      }
    };

    const getSeverityColor = (severity) => {
      switch (severity) {
        case 'critical': return 'bg-red-100 border-red-200';
        case 'warning': return 'bg-yellow-100 border-yellow-
200';
        default: return 'bg-blue-100 border-blue-200';
      }
    };

    return (
      <div className={`p-4 rounded-lg border
${getSeverityColor(alert.severity)} mb-2`}>
        <div className="flex items-start justify-between">
          <div className="flex items-center">
            {getAlertIcon(alert.severity)}
            <div className="ml-3">
              <h4 className="text-sm font-medium text-gray-
900">{alert.type}</h4>
              <p className="text-sm text-gray-
600">{alert.message}</p>
              <p className="text-xs text-gray-500 mt-1">
                {new Date(alert.timestamp).toLocaleString()} •
{alert.server}
              </p>
            </div>
          </div>
          <span className={`px-2 py-1 text-xs rounded-full ${
            alert.status === 'resolved' ? 'bg-green-100 text-
green-800' : 'bg-gray-100 text-gray-800'
          }`}>
            {alert.status}
          </span>
        </div>
      </div>
    );
  };

  // Дашборд
  const Dashboard = () => {
    const healthyServers = systemData.servers.filter(s =>
s.status === 'healthy').length;
    const totalAlerts = systemData.alerts.filter(a => a.status
=== 'active').length;
    const avgResponseTime = systemData.metrics.length > 0 ?
      Math.round(systemData.metrics.slice(-10).reduce((acc, m)
=> acc + m.responseTime, 0) / 10) : 0;
    const activeUsers = systemData.metrics.length > 0 ?
      Math.round(systemData.metrics[systemData.metrics.length -
1]?.activeUsers || 0) : 0;
```

```
    return (
      <div className="space-y-6">
        {/* Основні метрики */}
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-
cols-4 gap-6">
          <StatCard
            title="Здорові сервери"

value={`${healthyServers}/${systemData.servers.length}`}
            icon={Server}
            trend={5}
            color="#10B981"
          />
          <StatCard
            title="Активні алерти"
            value={totalAlerts}
            icon={AlertTriangle}
            trend={-12}
            color="#F59E0B"
          />
          <StatCard
            title="Час відгуку"
            value={`${avgResponseTime}ms`}
            icon={Clock}
            trend={-8}
            color="#3B82F6"
          />
          <StatCard
            title="Активні користувачі"
            value={activeUsers.toLocaleString()}
            icon={Users}
            trend={15}
            color="#8B5CF6"
          />
        </div>

        {/* Графіки продуктивності */}
        <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
          <div className="bg-white rounded-lg shadow-md p-6">
            <h3 className="text-lg font-semibold mb-
4">Використання ресурсів</h3>
            <ResponsiveContainer width="100%" height={300}>
              <LineChart data={systemData.metrics.slice(-20)}>
                <CartesianGrid strokeDasharray="3 3" />
                <XAxis dataKey="time" />
                <YAxis />
                <Tooltip />
                <Legend />
                <Line type="monotone" dataKey="cpu"
stroke="#EF4444" name="CPU %" />
                <Line type="monotone" dataKey="memory"
stroke="#3B82F6" name="RAM %" />
```

```
                <Line type="monotone" dataKey="disk"
stroke="#10B981" name="Диск %" />
              </LineChart>
            </ResponsiveContainer>
          </div>

          <div className="bg-white rounded-lg shadow-md p-6">
            <h3 className="text-lg font-semibold mb-4">Мережевий
трафік</h3>
            <ResponsiveContainer width="100%" height={300}>
              <AreaChart data={systemData.networkData.slice(-
12)}>
                <CartesianGrid strokeDasharray="3 3" />
                <XAxis dataKey="time" />
                <YAxis />
                <Tooltip />
                <Legend />
                <Area type="monotone" dataKey="inbound"
stackId="1" stroke="#3B82F6" fill="#93C5FD" name="Вхідний" />
                <Area type="monotone" dataKey="outbound"
stackId="1" stroke="#10B981" fill="#86EFAC" name="Вихідний" />
              </AreaChart>
            </ResponsiveContainer>
          </div>
        </div>

        {/* Статус сервісів та останні алерти */}
        <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
          <div className="bg-white rounded-lg shadow-md p-6">
            <h3 className="text-lg font-semibold mb-4">Статус
сервісів</h3>
            <div className="space-y-3">
              {systemData.services.slice(0, 5).map(service => (
                <div key={service.id} className="flex items-
center justify-between p-3 bg-gray-50 rounded-lg">
                  <div className="flex items-center">
                    <div className={`w-3 h-3 rounded-full mr-3
${
                      service.status === 'operational' ? 'bg-
green-500' :
                      service.status === 'degraded' ? 'bg-
yellow-500' : 'bg-red-500'
                    }`}></div>
                    <span className="font-
medium">{service.name}</span>
                  </div>
                  <span className="text-sm text-gray-
600">{service.responseTime.toFixed(0)}ms</span>
                </div>
              ))}
            </div>
          </div>
```

```
          <div className="bg-white rounded-lg shadow-md p-6">
            <h3 className="text-lg font-semibold mb-4">Останні
алерти</h3>
            <div className="max-h-80 overflow-y-auto">
              {systemData.alerts.slice(0, 5).map(alert => (
                <AlertItem key={alert.id} alert={alert} />
              ))}
            </div>
          </div>
        </div>
      </div>
    );
  };

  // Сторінка серверів
  const Servers = () => {
    const filteredServers = systemData.servers.filter(server =>
{
      const matchesSearch =
server.name.toLowerCase().includes(searchTerm.toLowerCase()) ||

server.type.toLowerCase().includes(searchTerm.toLowerCase());
      const matchesFilter = filterStatus === 'all' ||
server.status === filterStatus;
      return matchesSearch && matchesFilter;
    });

    return (
      <div className="space-y-6">
        {/* Фільтри та пошук */}
        <div className="bg-white rounded-lg shadow-md p-6">
          <div className="flex flex-col sm:flex-row gap-4">
            <div className="flex-1">
              <div className="relative">
                <Search className="absolute left-3 top-1/2
transform -translate-y-1/2 text-gray-400" size={20} />
                <input
                  type="text"
                  placeholder="Пошук серверів..."
                  value={searchTerm}
                  onChange={(e) =>
setSearchTerm(e.target.value)}
                  className="w-full pl-10 pr-4 py-2 border
border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500
focus:border-transparent"
                />
              </div>
            </div>
            <div className="flex gap-2">
              <select
                value={filterStatus}
                onChange={(e) =>
setFilterStatus(e.target.value)}
```

```
                className="px-4 py-2 border border-gray-300
rounded-lg focus:ring-2 focus:ring-blue-500"
              >
                <option value="all">Всі статуси</option>
                <option value="healthy">Здорові</option>
                <option value="warning">Попередження</option>
                <option value="critical">Критичні</option>
              </select>
            </div>
          </div>
        </div>

        {/* Список серверів */}
        <div className="grid grid-cols-1 lg:grid-cols-2 xl:grid-
cols-3 gap-6">
          {filteredServers.map(server => (
            <div key={server.id} className="bg-white rounded-lg
shadow-md p-6 border-l-4"
                 style={{
                   borderLeftColor:
                     server.status === 'healthy' ? '#10B981' :
                     server.status === 'warning' ? '#F59E0B' :
'#EF4444'
                 }}>
              <div className="flex items-center justify-between
mb-4">
                <h3 className="text-lg font-
semibold">{server.name}</h3>
                <span className={`px-2 py-1 text-xs rounded-full
${
                  server.status === 'healthy' ? 'bg-green-100
text-green-800' :
                  server.status === 'warning' ? 'bg-yellow-100
text-yellow-800' :
                  'bg-red-100 text-red-800'
                }`}>
                  {server.status}
                </span>
              </div>

              <div className="space-y-3">
                <div className="flex items-center justify-
between">
                  <span className="text-sm text-gray-
600">Тип:</span>
                  <span className="text-sm font-
medium">{server.type}</span>
                </div>
                <div className="flex items-center justify-
between">
                  <span className="text-sm text-gray-
600">Локація:</span>
                  <span className="text-sm font-
```

```
medium">{server.location}</span>
                </div>
                <div className="flex items-center justify-
between">
                  <span className="text-sm text-gray-
600">Uptime:</span>
                  <span className="text-sm font-
medium">{server.uptime} днів</span>
                </div>
              </div>

              <div className="mt-4 space-y-2">
                <div className="flex items-center justify-
between">
                  <span className="text-sm text-gray-
600">CPU</span>
                  <span className="text-sm font-
medium">{server.cpu.toFixed(1)}%</span>
                </div>
                <div className="w-full bg-gray-200 rounded-full
h-2">
                  <div
                    className="bg-blue-600 h-2 rounded-full"
                    style={{ width: `${server.cpu}%` }}
                  ></div>
                </div>

                <div className="flex items-center justify-
between">
                  <span className="text-sm text-gray-
600">RAM</span>
                  <span className="text-sm font-
medium">{server.memory.toFixed(1)}%</span>
                </div>
                <div className="w-full bg-gray-200 rounded-full
h-2">
                  <div
                    className="bg-green-600 h-2 rounded-full"
                    style={{ width: `${server.memory}%` }}
                  ></div>
                </div>

                <div className="flex items-center justify-
between">
                  <span className="text-sm text-gray-
600">Диск</span>
                  <span className="text-sm font-
medium">{server.disk.toFixed(1)}%</span>
                </div>
                <div className="w-full bg-gray-200 rounded-full
h-2">
                  <div
                    className="bg-purple-600 h-2 rounded-full"
```

```
                    style={{ width: `${server.disk}%` }}
                  ></div>
                </div>
              </div>
            </div>
          ))}
        </div>
      </div>
    );
  };

  // Сторінка аналітики
  const Analytics = () => {
    const performanceByMetric =
systemData.performanceData.reduce((acc, item) => {
      if (!acc[item.metric]) {
        acc[item.metric] = [];
      }
      acc[item.metric].push(item);
      return acc;
    }, {});

    const errorDistribution = systemData.alerts.reduce((acc,
alert) => {
      acc[alert.severity] = (acc[alert.severity] || 0) + 1;
      return acc;
    }, {});

    const errorData =
Object.entries(errorDistribution).map(([severity, count]) => ({
      name: severity,
      value: count,
      color: severity === 'critical' ? '#EF4444' : severity ===
'warning' ? '#F59E0B' : '#3B82F6'
    }));

    return (
      <div className="space-y-6">
        {/* Детальна аналітика продуктивності */}
        <div className="bg-white rounded-lg shadow-md p-6">
          <h3 className="text-lg font-semibold mb-4">Тренды
продуктивності</h3>
          <div className="grid grid-cols-1 lg:grid-cols-2 gap-
6">
            {Object.entries(performanceByMetric).map(([metric,
data]) => (
              <div key={metric}>
                <h4 className="text-md font-medium mb-
2">{metric}</h4>
                <ResponsiveContainer width="100%" height={200}>
                  <LineChart data={data.slice(-12)}>
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="time" />
```

```
                    <YAxis />
                    <Tooltip />
                    <Line type="monotone" dataKey="value"
stroke="#3B82F6" strokeWidth={2} />
                  </LineChart>
                </ResponsiveContainer>
              </div>
            ))}
          </div>
        </div>

        {/* Розподіл помилок та час відгуку */}
        <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
          <div className="bg-white rounded-lg shadow-md p-6">
            <h3 className="text-lg font-semibold mb-4">Розподіл
алертів за важливістю</h3>
            <ResponsiveContainer width="100%" height={300}>
              <PieChart>
                <Pie
                  data={errorData}
                  cx="50%"
                  cy="50%"
                  labelLine={false}
                  label={({name, percent}) => `${name}
${(percent * 100).toFixed(0)}%`}
                  outerRadius={80}
                  fill="#8884d8"
                  dataKey="value"
                >
                  {errorData.map((entry, index) => (
                    <Cell key={`cell-${index}`}
fill={entry.color} />
                  ))}
                </Pie>
                <Tooltip />
              </PieChart>
            </ResponsiveContainer>
          </div>

          <div className="bg-white rounded-lg shadow-md p-6">
            <h3 className="text-lg font-semibold mb-4">Час
відгуку по годинах</h3>
            <ResponsiveContainer width="100%" height={300}>
              <BarChart data={systemData.metrics.slice(-24)}>
                <CartesianGrid strokeDasharray="3 3" />
                <XAxis dataKey="time" />
                <YAxis />
                <Tooltip />
                <Bar dataKey="responseTime" fill="#3B82F6" />
              </BarChart>
            </ResponsiveContainer>
          </div>
        </div>
```

```
        {/* Статистика помилок */}
        <div className="bg-white rounded-lg shadow-md p-6">
          <h3 className="text-lg font-semibold mb-4">Кількість
помилок в часі</h3>
          <ResponsiveContainer width="100%" height={300}>
            <AreaChart data={systemData.metrics.slice(-20)}>
              <CartesianGrid strokeDasharray="3 3" />
              <XAxis dataKey="time" />
              <YAxis />
              <Tooltip />
              <Legend />
              <Area type="monotone" dataKey="errors"
stroke="#EF4444" fill="#FEE2E2" name="Помилки" />
            </AreaChart>
          </ResponsiveContainer>
        </div>
      </div>
    );
  };

  // Сторінка алертів
  const Alerts = () => {
    const filteredAlerts = systemData.alerts.filter(alert => {
      const matchesSearch =
alert.type.toLowerCase().includes(searchTerm.toLowerCase()) ||

alert.message.toLowerCase().includes(searchTerm.toLowerCase());
      const matchesFilter = filterStatus === 'all' ||
alert.severity === filterStatus;
      return matchesSearch && matchesFilter;
    });

    return (
      <div className="space-y-6">
        {/* Фільтри */}
        <div className="bg-white rounded-lg shadow-md p-6">
          <div className="flex flex-col sm:flex-row gap-4">
            <div className="flex-1">
              <div className="relative">
                <Search className="absolute left-3 top-1/2
transform -translate-y-1/2 text-gray-400" size={20} />
                <input
                  type="text"
                  placeholder="Пошук алертів..."
                  value={searchTerm}
                  onChange={(e) =>
setSearchTerm(e.target.value)}
                  className="w-full pl-10 pr-4 py-2 border
border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500
focus:border-transparent"
                />
              </div>
```

```
            </div>
            <div className="flex gap-2">
              <select
                value={filterStatus}
                onChange={(e) =>
setFilterStatus(e.target.value)}
                className="px-4 py-2 border border-gray-300
rounded-lg focus:ring-2 focus:ring-blue-500"
              >
                <option value="all">Всі рівні</option>
                <option value="critical">Критичні</option>
                <option value="warning">Попередження</option>
                <option value="info">Інформаційні</option>
              </select>
            </div>
          </div>
        </div>

        {/* Список алертів */}
        <div className="bg-white rounded-lg shadow-md p-6">
          <h3 className="text-lg font-semibold mb-4">Всі
алерти</h3>
          <div className="space-y-3">
            {filteredAlerts.map(alert => (
              <AlertItem key={alert.id} alert={alert} />
            ))}
          </div>
        </div>
      </div>
    );
  };

  // Головна навігація
  const Navigation = () => (
    <nav className="bg-white shadow-sm border-b">
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div className="flex justify-between h-16">
          <div className="flex items-center">
            <Activity className="h-8 w-8 text-blue-600 mr-3" />
            <h1 className="text-xl font-bold text-gray-
900">Система моніторингу</h1>
          </div>

          <div className="flex items-center space-x-4">
            <div className="flex items-center space-x-2">
              <Clock size={16} className="text-gray-500" />
              <select
                value={selectedTimeRange}
                onChange={(e) =>
setSelectedTimeRange(e.target.value)}
                className="text-sm border border-gray-300
rounded px-2 py-1"
              >
```

```
                <option value="1h">Остання година</option>
                <option value="6h">Останні 6 годин</option>
                <option value="24h">Остання доба</option>
                <option value="7d">Останній тиждень</option>
              </select>
            </div>

            <button
              onClick={() => setIsRealTime(!isRealTime)}
              className={`flex items-center space-x-1 px-3 py-1
rounded-full text-sm ${
                isRealTime ? 'bg-green-100 text-green-800' :
'bg-gray-100 text-gray-800'
              }`}
            >
              <RefreshCw size={14} className={isRealTime ?
'animate-spin' : ''} />
              <span>{isRealTime ? 'Реальний час' :
'Призупинено'}</span>
            </button>

            <button className="p-2 text-gray-500 hover:text-
gray-700">
              <Bell size={20} />
              {notifications.length > 0 && (
                <span className="absolute -mt-2 -mr-2 bg-red-500
text-white text-xs rounded-full h-5 w-5 flex items-center
justify-center">
                  {notifications.length}
                </span>
              )}
            </button>

            <button className="p-2 text-gray-500 hover:text-
gray-700">
              <Settings size={20} />
            </button>
          </div>
        </div>
      </div>
    </nav>
  );

  // Таби навігації
  const TabNavigation = () => (
    <div className="bg-white shadow-sm">
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div className="flex space-x-8">
          {[
            { id: 'dashboard', name: 'Дашборд', icon: BarChart3
},
            { id: 'servers', name: 'Сервери', icon: Server },
            { id: 'analytics', name: 'Аналітика', icon:
```

```
TrendingUp },
            { id: 'alerts', name: 'Алерти', icon: AlertTriangle
}
          ].map(tab => (
            <button
              key={tab.id}
              onClick={() => setActiveTab(tab.id)}
              className={`flex items-center space-x-2 py-4 px-1
border-b-2 font-medium text-sm ${
                activeTab === tab.id
                  ? 'border-blue-500 text-blue-600'
                  : 'border-transparent text-gray-500
hover:text-gray-700 hover:border-gray-300'
              }`}
            >
              <tab.icon size={16} />
              <span>{tab.name}</span>
            </button>
          ))}
        </div>
      </div>
    </div>
  );

  return (
    <div className="min-h-screen bg-gray-50">
      <Navigation />
      <TabNavigation />

      <main className="max-w-7xl mx-auto py-6 px-4 sm:px-6
lg:px-8">
        {activeTab === 'dashboard' && <Dashboard />}
        {activeTab === 'servers' && <Servers />}
        {activeTab === 'analytics' && <Analytics />}
        {activeTab === 'alerts' && <Alerts />}
      </main>
    </div>
  );
};

export default MonitoringSystem;
```