ДОДАТОК А

Код програми

```
<!DOCTYPE html>
<html>

<head>
  <link rel="stylesheet" href="https://unpkg.com/bulma@0.9.2/css/bulma.min.css">
  <style>
    body {
      background-color: #222;
      color: #fff;
      font-family: 'Arial', sans-serif;
    }

    .title {
      color: #ffffff;
      font-size: 2rem;
      font-weight: 600;
      line-height: 1.125;
    }

    .custom-table-container .table-container {
      border-radius: 10px;
      overflow: hidden;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.3);
      margin-bottom: 20px;
```

```
  background-color: rgb(95, 95, 95) !important;
}


.table-container table {
  border-radius: 10px;
  overflow: hidden;
}

.table-container th,
.table-container td {
  border-radius: 0;
  text-align: center;
  color: inherit;
  /* Змінено колір тексту на успадкований колір */
  cursor: pointer;
  /* Додано курсор-покажчик */
  position: sticky;
  /* Додано позиціонування sticky */
  top: 0;
  /* Додано верхню позицію 0 */
  background-color: #fff;
  /* Додано білий фон */
}

.table-container th.sort-asc::after,
.table-container th.sort-desc::after {
  content: ' ▼';
```

```
  opacity: 0.3;
  margin-left: 5px;
  color: inherit;
  /* Додано стиль 'color: inherit;' */
}

.table-container th.sort-desc::after {
  content: ' ▲';
  /* Додано символ стрілки вверх для сортування за зростанням */
}

.table-container th.sorted {
  color: #000000;
}

.chart-container {
  border-radius: 10px;
  overflow: hidden;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.3);
  margin-bottom: 20px;
}

.pagination {
  justify-content: center;
}

.status-container {
  border-radius: 10px;
```

```
    overflow: hidden;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.3);
    margin-bottom: 20px;
    padding: 20px;
    background-color: white;
    /* Додано білий фон */
   }

   .status-container p {
    font-weight: bold;
    font-size: 18px;
    color: green;
    /* Зелений колір тексту */
   }

   .status-container.closed p {
    color: red;
    /* Червоний колір тексту */
   }
  </style>
  <script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/chart.js@2.9.4/dist/Chart.bundle.min.js"></script>
</head>

<body>
  <div id="app" class="container">
   <h1 class="title">Дипломний проект з моніторингу данних Черноморченко
Б.О.</h1>
```

```
<h2 class="title" style="text-align: center;">Данні з датчиків</h2>
<div class="table-container custom-table-container">
 <table class="table is-fullwidth is-striped">
  <thead>
   <tr>
    <th @click="sortBy('temperature')"
     :class="{'sort-asc': sortKey === 'temperature' && sortDirection === 'asc', 'sort-desc': sortKey === 'temperature' && sortDirection === 'desc', 'sorted': sortKey === 'temperature'}">
     Температура (°C)</th>
    <th @click="sortBy('humidity')"
     :class="{'sort-asc': sortKey === 'humidity' && sortDirection === 'asc', 'sort-desc': sortKey === 'humidity' && sortDirection === 'desc', 'sorted': sortKey === 'humidity'}">
     Вологість (%)</th>
    <th @click="sortBy('time')"
     :class="{'sort-asc': sortKey === 'time' && sortDirection === 'asc', 'sort-desc': sortKey === 'time' && sortDirection === 'desc', 'sorted': sortKey === 'time'}">
     Час (рррр/мм/дд/год/хв/сек)</th>
   </tr>
  </thead>
  <tbody>
   <tr v-for="(item, index) in paginatedData" :key="index">
    <td>{{ item.temperature }}</td>
    <td>{{ item.humidity }}</td>
    <td>{{ item.time }}</td>
   </tr>
  </tbody>
 </table>
</div>
```

```
    <div class="status-container" :class="{'closed': getLastContactStatus() === 'Закрито'}">
      <h2 class="subtitle" style="text-align: center;">Стан дверей:</h2>
      <p style="text-align: center;">{{ getLastContactStatus() }}</p>
    </div>

    <nav class="pagination is-centered">
      <a class="pagination-previous" @click="previousPage" :disabled="currentPage === 1">Попередня сторінка</a>
      <a class="pagination-next" @click="nextPage" :disabled="currentPage === totalPages">Наступна сторінка</a>
    </nav>

    <div class="chart-container" style="width: 50%; float: left;">
      <!-- Змінено ширину та додано float: left; -->
      <h2 class="subtitle">Графік температури</h2>
      <canvas id="temperatureChart"></canvas>
    </div>

    <div class="chart-container" style="width: 50%; float: right;">
      <!-- Змінено ширину та додано float: right; -->
      <h2 class="subtitle">Графік вологості</h2>
      <canvas id="humidityChart"></canvas>
    </div>
  </div>
  <script>
    // Підключення WebSocket на стороні клієнта
    const socket = new WebSocket('ws://localhost:8080/ws');
    socket.onclose = function (event) {
```

```
    if (event.code === 1001) {
      console.log('WebSocket connection closed by client: going away.');
      // Обробити закриття з'єднання відповідним чином, наприклад, очистити, відписатися тощо.
    } else {
      console.log('WebSocket connection closed with code:', event.code);
      // Обробити інші коди закриття за необхідності.
    }
  };


  // Обробник події відкриття з'єднання WebSocket
  socket.onopen = function () {
    console.log('WebSocket connection established.');
  };

  // Обробник події отримання повідомлення через WebSocket
  socket.onmessage = function (event) {
    const data = JSON.parse(event.data);
    // Оновлення відповідних елементів на сторінці з отриманими даними
    updatePageWithData(data);
    // Перезавантажити сторінку
    location.reload();
  };

  // Функція для оновлення відповідних елементів на сторінці
  function updatePageWithData(data) {
    // Ваш код для оновлення відповідних елементів на сторінці з отриманими даними
```

```
}

// Отримання даних з API
async function getData() {
  const response = await fetch('http://localhost:8080/api/data');
  const data = await response.json();
  return data;
}

// Ініціалізація Vue.js
new Vue({
  el: '#app',
  data: {
    data: [],
    currentPage: 1,
    pageSize: 10,
    sortKey: 'time',
    sortDirection: 'desc',
    socket: null,
  },
  methods: {

    formatDateTime(dateTimeString) {
      const date = new Date(dateTimeString);
      return date.toLocaleTimeString('uk-UA');
      },

    // Обробка повідомлень веб-сокету
```

```
  handleWebSocketMessage(event) {
    const newData = JSON.parse(event.data);
    this.data.push(newData);
    this.updateCharts();
  },
},
async mounted() {
  // Отримання даних при завантаженні сторінки
  this.data = await getData();

  // Підключення до веб-сокету
  this.socket = new WebSocket('ws://localhost:8080/ws');
  this.socket.addEventListener('message', this.handleWebSocketMessage);
},
computed: {
  paginatedData() {
    const startIndex = (this.currentPage - 1) * this.pageSize;
    const endIndex = startIndex + this.pageSize;
    const sortedData = this.sortData(this.data);
    return sortedData.slice(startIndex, endIndex);
  },
  totalPages() {
    return Math.ceil(this.data.length / this.pageSize);
  },
},
async mounted() {
  // Отримання даних при завантаженні сторінки
  this.data = await getData();
```

```
// Змінено this.data на this.data.slice(-25), щоб обмежити дані до останніх 25 значень


// Створення графіку температури
const temperatureChart = new Chart(document.getElementById('temperatureChart'), {
  type: 'line',
  data: {
    labels: this.data.map(item => item.time),
    datasets: [{
      label: 'Температура',
      data: this.data.map(item => item.temperature),
      borderColor: 'red',
      backgroundColor: 'rgba(255, 0, 0, 0.3)',
      borderWidth: 2,
      pointRadius: 0,
    }],
  },
  options: {
    responsive: true,
    maintainAspectRatio: false,
    scales: {
      x: {
        type: 'time',
        time: {
          displayFormats: {
            hour: 'H:mm',
            day: 'MMM D',
            week: 'll',
```

```
          month: 'MMM YYYY',
        },
      },
      ticks: {
        fontColor: '#fff',
      },
      grid: {
        color: 'rgba(255, 255, 255, 0.1)',
      },
    },
    y: {
      ticks: {
        fontColor: '#fff',
      },
      grid: {
        color: 'rgba(255, 255, 255, 0.1)',
      },
      stepSize: 1, // Зменшуємо відстань між точками по осі y
    },
  },
},
});

// Створення графіку вологості
const humidityChart = new Chart(document.getElementById('humidityChart'), {
  type: 'line',
  data: {
    labels: this.data.map(item => item.time),
```

```
  datasets: [{
    label: 'Вологость',
    data: this.data.map(item => item.humidity),
    borderColor: 'blue',
    backgroundColor: 'rgba(0, 0, 255, 0.3)',
    borderWidth: 2,
    pointRadius: 0,
  }],
},
options: {
  responsive: true,
  maintainAspectRatio: false,
  scales: {
    x: {
      type: 'time',
      time: {
        displayFormats: {
          hour: 'H:mm',
          day: 'MMM D',
          week: 'll',
          month: 'MMM YYYY',
        },
      },
      ticks: {
        fontColor: '#fff',
      },
      grid: {
        color: 'rgba(255, 255, 255, 0.1)',
```

```
        },
      },
      y: {
        ticks: {
          fontColor: '#fff',
        },
        grid: {
          color: 'rgba(255, 255, 255, 0.1)',
        },
        stepSize: 1, // Зменшуємо відстань між точками по осі y
      },
    },
  },
 });
},
methods: {
 previousPage() {
  if (this.currentPage > 1) {
   this.currentPage--;
  }
 },
 nextPage() {
  if (this.currentPage < this.totalPages) {
   this.currentPage++;
  }
 },
 getLastContactStatus() {
  const lastData = this.data[this.data.length - 1];
```

```
      if (lastData && lastData.contact) {
        return 'Закрито';
      } else {
        return 'Відкрито';
      }
    },
    formatContact(contact) {
      if (contact) {
        return 'Закрито';
      } else {
        return 'Відкрито';
      }
    },
    sortBy(key) {
      if (this.sortKey === key) {
        this.sortDirection = this.sortDirection === 'asc' ? 'desc' : 'asc';
      } else {
        this.sortKey = key;
        this.sortDirection = 'asc';
      }
      this.updateCharts(); // Оновлюємо графіки після сортування
    },
    updateCharts() {
      const temperatureChart = this.getTemperatureChartInstance();
      const humidityChart = this.getHumidityChartInstance();
      temperatureChart.data.labels = this.data.map(item => item.time);
      temperatureChart.data.datasets[0].data = this.data.map(item =>
item.temperature);
```

```
    humidityChart.data.labels = this.data.map(item => item.time);
    humidityChart.data.datasets[0].data = this.data.map(item => item.humidity);
    temperatureChart.update();
    humidityChart.update();
  },
  getTemperatureChartInstance() {
    return Chart.getChart('temperatureChart');
  },
  getHumidityChartInstance() {
    return Chart.getChart('humidityChart');
  },
  sortData(data) {
    const key = this.sortKey;
    const direction = this.sortDirection === 'asc' ? 1 : -1;
    return data.slice().sort((a, b) => {
      const valueA = a[key];
      const valueB = b[key];
      if (valueA < valueB) {
        return -1 * direction;
      } else if (valueA > valueB) {
        return 1 * direction;
      } else {
        return 0;
      }
    });
  },
 },
});
```

```
  </script>
</body>

</html>
```

ДОДАТОК Б

Демонстраційний матеріал

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 502900.002 ПЗ	Пояснювальна записка	А4, 74 с.
		Додаткові матеріали	
2		Код програми	А4, 16 с.
3		Демонстраційний матеріал	А4, 7 с.

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК. 502900.002 ВД		
Розроб.		Бірюков О.О.			*Розробка системи автоматизації для моніторингу з використанням бездротових мереж* Відомість бакалаврської роботи	Літера	Аркуш	Аркушів
Перевір.		Стародубцев М.Г.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Бронніков А. І.						
Затв.		Невлюдов І.Ш.						