

Laporan Praktikum

Mata Kuliah Pemrograman Web & Pemrograman Berorientasi Objek



Pertemuan 16

"Menghubungkan WebSocket dengan Database MySQL"

Dosen Pengampu :

Willdan Aprizal Arifin S.pd M.kom

Disusun Oleh :

RIYADI TRI WALUYA BUDI (2308065)

Angga Ardiansyah (2307300)

PROGRAM STUDI SISTEM INFORMASI KELAUTAN

UNIVERSITAS PENDIDIKAN INDONESIA

2024

I. PENDAHULUAN

Di era digital yang semakin berkembang, kebutuhan akan komunikasi real-time dalam aplikasi web telah menjadi suatu kebutuhan. Teknologi WebSocket hadir sebagai solusi yang memungkinkan komunikasi dua arah secara simultan antara klien dan server, mengatasi keterbatasan protokol HTTP tradisional yang bersifat request-response. Namun, ketika dipadukan dengan sistem manajemen basis data seperti MySQL, WebSocket dapat menciptakan aplikasi web yang dinamis dan responsif.

WebSocket adalah protokol komunikasi yang memungkinkan interaksi dua arah secara real-time antara klien dan server melalui satu koneksi TCP. Teknologi ini sering digunakan pada aplikasi berbasis web yang memerlukan update data secara real-time seperti aplikasi chatting, notifikasi, tabel data dinamis. MySQL adalah sistem manajemen basis data relasional (RDBMS) yang banyak digunakan untuk menyimpan dan mengelola data aplikasi. Dalam skenario dunia nyata, data dikirim dari klien ke server melalui WebSocket, dan WebSocket menyimpan atau mengambil data ke database MySQL.

Melalui praktikum ini, mahasiswa akan mempelajari cara membangun aplikasi web sederhana yang mengintegrasikan WebSocket dengan MySQL. Fokus utama adalah memahami konsep dasar WebSocket, cara implementasinya menggunakan teknologi web modern, serta bagaimana menghubungkannya dengan basis data MySQL untuk menciptakan aplikasi yang interaktif dan responsif.

II. ALAT DAN BAHAN

1. Laptop/Komputer
2. Visual Studio Code
3. Node.js (dengan library ws atau socket.io)
4. MySQL

III. LANGKAH KERJA

1. Lakukan instalasi pada terminal dengan menggunakan:

```
npm init -y  
npm install ws mysql
```

2. Buat server menggunakan ws (Websocket library).

```
const WebSocket = require("ws");
const db = require("./db");

const wss = new WebSocket.Server({ port: 3000 });

wss.on("connection", (ws) => {
    console.log("Client connected");

    // Kirim data dari MySQL ke klien saat koneksi terhubung
    db.query("SELECT * FROM logs", (err, results) => {
        if (err) {
            console.error("Gagal membaca data dari database:", err);
            return;
        }
        ws.send(JSON.stringify({ type: "initial_data", data: results }));
    });

    // Kirim pesan ke klien setiap 3 detik
    setInterval(() => {
        ws.send(
            JSON.stringify({ message: "Data dari server", timestamp: new Date() })
    });
}, 3000);

// Terima pesan dari klien
ws.on("message", (message) => {
    console.log(`Pesan dari client: ${message}`);

    // Sistem notifikasi jika terdapat keyword spesifik
    const keyword = "penting";
    if (message.includes(keyword)) {
        ws.send(
            JSON.stringify({ type: "notification", message: "Keyword penting terdeteksi!" })
        );
    }
});
```

```

// Kirim notifikasi ke semua klien
wss.clients.forEach((client) => {
    if (client !== ws && client.readyState === WebSocket.OPEN) {
        client.send(
            JSON.stringify({ type: "notification", message: `Notifikasi:
keyword ${keyword} ditemukan.` })
        );
    }
});

// Simpan pesan ke database
const query = "INSERT INTO logs (message, timestamp) VALUES (?, ?)";
const values = [message, new Date()];
db.query(query, values, (err, result) => {
    if (err) {
        console.error("Gagal menyimpan data ke MySQL:", err);
    } else {
        console.log("Data berhasil disimpan ke database, ID:",
result.insertId);
    }
});

ws.on("close", () => {
    console.log("Client disconnected");
});

console.log("WebSocket server berjalan di ws://localhost:3000");

```

- Buat database dan table.

```

CREATE TABLE `logs` (
`id` int(11) NOT NULL,
`message` varchar(255) DEFAULT NULL,
`timestamp` datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

```

- Buat db.js untuk mengelola koneksi dan operasi database.

```
const mysql = require("mysql");
```

```

const db = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "websocket_db",
});

db.connect((err) => {
  if (err) {
    console.error("Error koneksi MySQL:", err);
  } else {
    console.log("Terhubung ke MySQL");
  }
});

module.exports = db;

```

5. Buat app.js untuk mengatur struktur dasar aplikasi, menginisialisasi modul atau middleware, dan menjalankan server.

```

const WebSocket = require("ws");
const db = require("./db");

// Koneksi ke WebSocket Server
const ws = new WebSocket("ws://localhost:3000");

ws.on("open", () => {
  console.log("Terhubung ke WebSocket server");

  // Kirim pesan ke server
  ws.send("Halo Server! Ini dari Client");
});

// Terima data dari WebSocket server
ws.on("message", (data) => {
  console.log("Data diterima dari server:", data);

  const jsonData = JSON.parse(data);
  if (jsonData.type === "notification") {
    console.log("Notifikasi diterima:", jsonData.message);
  } else if (jsonData.type === "initial_data") {
    console.log("Data awal dari server:", jsonData.data);
  } else {
    // Simpan data ke MySQL
  }
});

```

```

const query = "INSERT INTO logs (message, timestamp) VALUES (?, ?)";
const values = [jsonData.message, jsonData.timestamp];
db.query(query, values, (err, result) => {
  if (err) {
    console.error("Gagal menyimpan data ke MySQL:", err);
  } else {
    console.log("Data berhasil disimpan ke database, ID:", result.insertId);
  }
});

// Handle error
ws.on("error", (err) => {
  console.error("Error WebSocket:", err);
});

ws.on("close", () => {
  console.log("Koneksi WebSocket ditutup");
});

```

6. Buat index.html untuk frontend nya.

```

<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>WebSocket Frontend</title>
    <!-- Bootstrap CSS -->
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css"
      rel="stylesheet">
  </head>

  <body>
    <div class="container my-5">
      <h1 class="text-center mb-4">WebSocket Frontend</h1>

      <div id="messages" class="border rounded p-3 mb-3 bg-light"
        style="height: 300px; overflow-y: auto;">
        <p class="text-muted text-center">Belum ada pesan</p>
      </div>
    </div>
  </body>

```

```
<div class="input-group">
    <input type="text" id="inputMessage" class="form-control"
placeholder="Ketik pesan...">
    <button class="btn btn-primary"
onclick="sendMessage()">Kirim</button>
</div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"></script>
<script>
    const ws = new WebSocket("ws://localhost:3000");
    const messagesDiv = document.getElementById("messages");

    ws.onopen = () => {
        console.log("Terhubung ke WebSocket server");
    };

    ws.onmessage = (event) => {
        const data = JSON.parse(event.data);

        if (data.type === "notification") {

            alert(data.message);
            const notification = document.createElement("div");
            notification.className = "alert alert-info";
            notification.textContent = "📢 " + data.message;
            messagesDiv.appendChild(notification);
        } else if (data.type === "initial_data") {

            data.data.forEach((log) => {
                const logMessage = document.createElement("p");
                logMessage.className = "text-secondary";
                logMessage.textContent = `[${log.timestamp}]
${log.message}`;
                messagesDiv.appendChild(logMessage);
            });
        } else {

            const message = document.createElement("p");
            message.className = "text-primary";
            message.textContent = data.message || "Data diterima: " +
event.data;
        }
    };
</script>
```

```

        messagesDiv.appendChild(message);
    }

    messagesDiv.scrollTop = messagesDiv.scrollHeight;
};

function sendMessage() {
    const input = document.getElementById("inputMessage");
    const message = input.value.trim();

    if (message) {
        ws.send(message);
        input.value = "";
    } else {
        alert("Pesan tidak boleh kosong!");
    }
}

ws.onerror = (err) => {
    console.error("WebSocket error:", err);
};

ws.onclose = () => {
    console.log("Koneksi WebSocket ditutup");
};

</script>
</body>

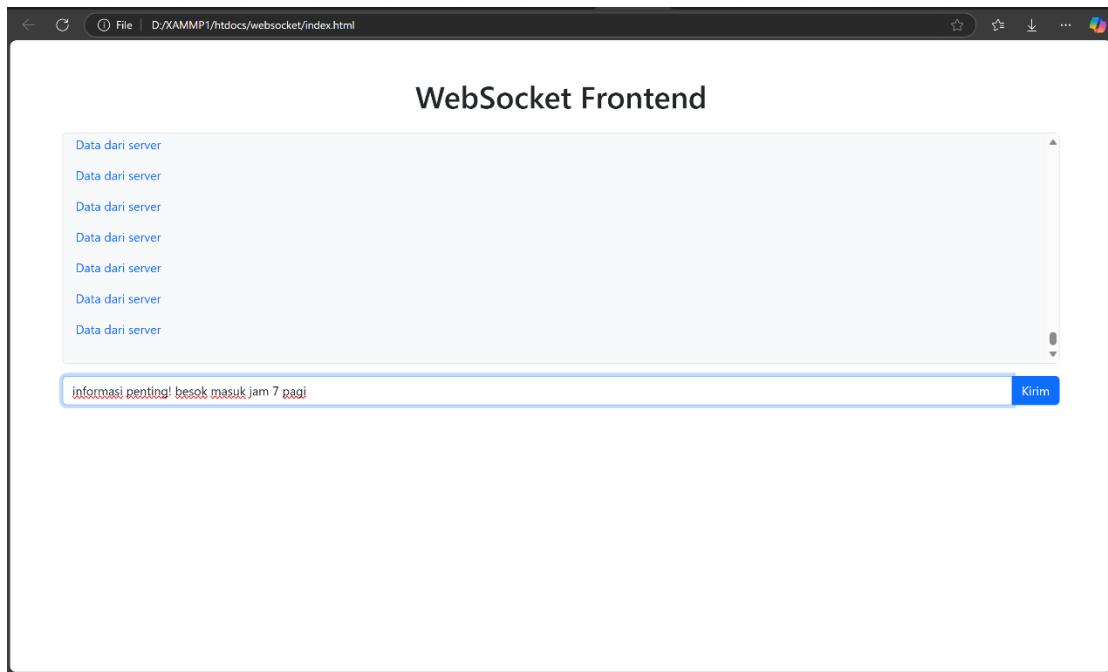
</html>

```

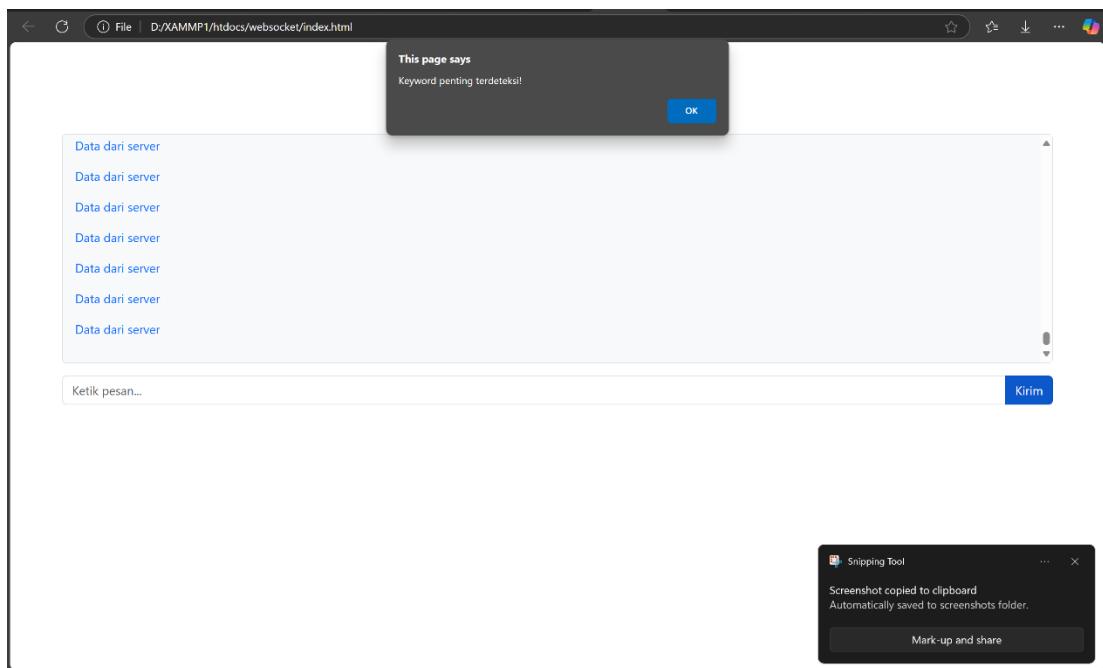
7. Lakukan pengujian aplikasi.
 - a. Jalankan server websocket dengan menggunakan:
`node server.js`
 - b. Jalankan client websocket dengan menggunakan:
`node app.js`

Hasil Akhir :

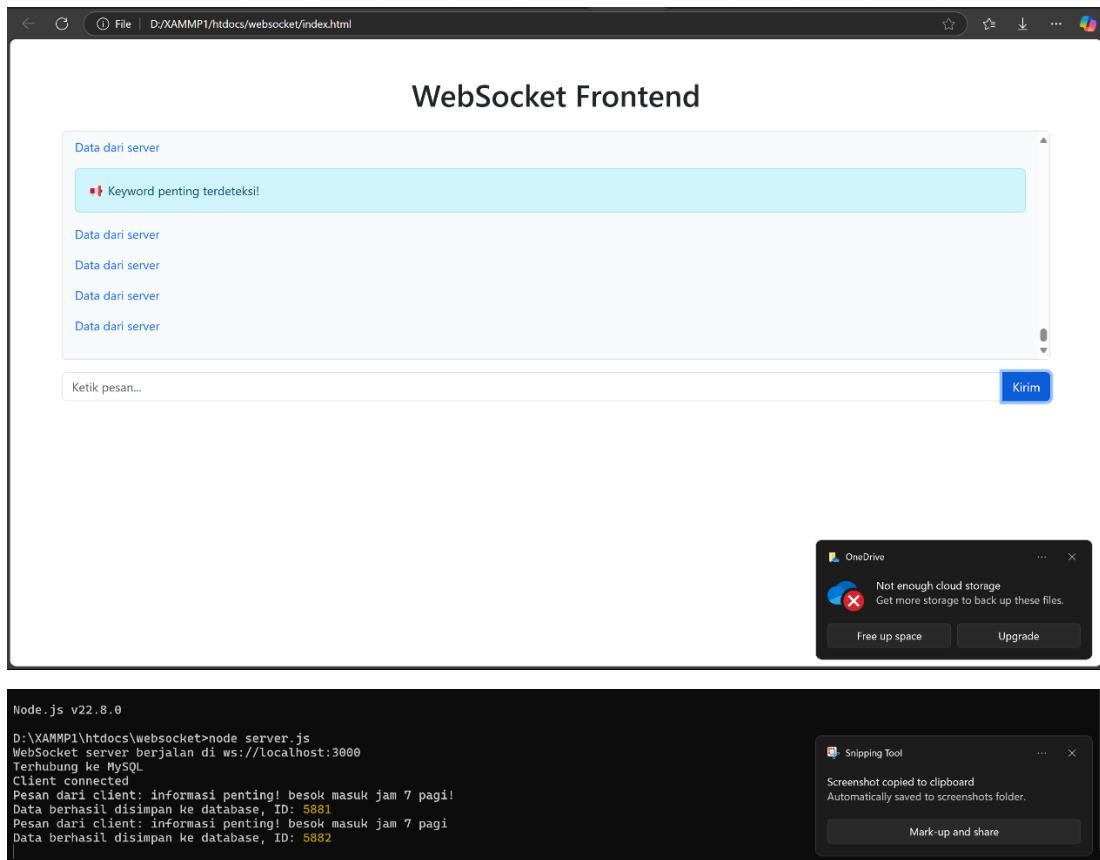
1. Tampilan website sederhana.



2. Alert ketika keyword “penting” terdeteksi.



3. Pesan tersimpan pada database.



Link GitHub: https://github.com/anggard4n/k_websocket

IV. KESIMPULAN

Melalui praktikum ini menambah pemahaman mahasiswa mengenai menghubungkan WebSocket dengan database MySQL dapat memungkinkan komunikasi real-time yang efisien antara server dan klien. Dalam praktiknya, WebSocket digunakan untuk mempertahankan koneksi dua arah yang terus aktif, sementara MySQL bertindak sebagai penyimpanan data yang andal. Melalui pengelolaan koneksi yang tepat, data dari database dapat dikirim dan diterima secara langsung tanpa perlu penyegaran halaman, sehingga meningkatkan responsivitas aplikasi. Implementasi ini membutuhkan pemahaman tentang manajemen koneksi WebSocket, pengelolaan query MySQL, serta keamanan data selama proses komunikasi.