TUGAS JOBSHEET 9

SOCKET PROGRAMMING

PRAKTIKUM PEMROGRAMAN JARINGAN

Dosen Pengampu:

Randi Proska Sandra, S.Pd, M.Sc.

Disusun oleh:

Muhammad Ghazian Tsaqif Zhafiri Andoz

(23343057)

Seksi:

202513430094

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK ELEKTRONIKA

UNIVERSITAS NEGERI PADANG

202

1. Jelaskan dengan disertai penjelasan baris kode terkait perbedaan fungsi socket.on yang ada pada file index.js di folder src dan file chat.js pada folder public/js.

Aspek	socket.on di src/index.js (Server)	socket.on di public/js/chat.js (Klien)
Lingkungan eksekusi	Node.js	Browser
Tujuan utama	Memproses event dari klien. Mengelola user. Mengirim pesan ke room.	Menangkap event dari server. Menampilkan data ke UI.
Contoh event	join, kirimPesan, kirimLokasi, disconnect	pesan, locationMessage, roomData
Fungsi dasar	Menerima data dari klien dan menjalankan logika server	Menerima data dari server dan merender ke tampilan
Contoh baris kode	`socket.on("kirimPesan", (pesan, callback) => {...})`	`socket.on("pesan", (message) => {...})`
Aksi yang dilakukan	Validasi pesan, broadcast, simpan user, update room	Render Mustache, update sidebar, autoScroll
Data yang diterima	Data mentah dari input pengguna	Data yang sudah diproses server
Output	Emit event ke semua klien dalam room	Ubah HTML, tampilkan pesan, tampilkan link lokasi

2. Pada saat anda melakukan proses chat seperti pada langkah 12 dan 13. Bukalah Inspect pada browser anda lalu buka menu Console. Lakukan proses chat dan investigasi apa yang ditampilkan pada console tersebut. Uraikan penjelasan anda dengan mengaitkannya ke baris kode yang menurut anda berhubungan dengan hal tersebut.

Tampilan di Console	Penyebab muncul	Baris kode yang berhubungan
Objek pesan seperti: `{username: "Admin", text: "Selamat datang!", createdAt: ...}`	Klien menerima event pesan dari server lalu mencetak isi event ke console	`js socket.on("pesan", (message) => { console.log(message); })`
Objek pesan seperti: `{username: "randi`	Server melakukan broadcast pesan kembali ke semua	`js socket.on("pesan", (message) => {`

proska", text: "test", createdAt: ...}	klien dalam room termasuk pengirim. Klien menangkap event itu dan mencetaknya	*console.log(message); })*
Teks Pesan berhasil dikirim	Callback dari kirimPesan dieksekusi setelah server menerima dan mem-broadcast pesan	*js console.log("Pesan berhasil dikirim");* pada bagian:
js socket.emit("kirimPesan", pesan, (error) => {...})		
Teks atau objek lain muncul setiap pesan dikirim	Setiap event yang diterima dari server memiliki console.log() di handler sehingga semua data yang dipancarkan server tercatat di console	Semua blok socket.on(...) yang memiliki console.log()

Alur proses yang menyebabkan log tersebut
1. Klien masuk ke room. Server mengirim pesan selamat datang melalui event pesan. Klien mencetak objek pesan dari event tersebut. 2. Klien mengirim pesan melalui socket.emit("kirimPesan"). Setelah server menerima, callback dijalankan sehingga klien mencetak Pesan berhasil dikirim. 3. Server mem-broadcast pesan itu ke seluruh klien dalam room. Klien menerima event pesan dan mencetak objek pesan yang diterima di console. 4. Setiap pesan berikutnya mengikuti pola yang sama sehingga console berisi urutan objek pesan lengkap sesuai aktivitas chat.

3. Pada file chat.html di bagian akhir pada baris kode <script> terdapat penggunaan library Mustache, Moment, dan Qs. Jelaskan bagaimana ketiga library ini berfungsi dalam aplikasi yang anda buat. Kaitkan dengan baris kode yang menurut anda berhubungan.
   - **Mustache** → Digunakan untuk merender template HTML dinamis berdasarkan data yang diterima dari server. Mustache membaca template di <script id="message-template">, <script id="locationMessage-template">, dan <script id="sidebar-template"> lalu mengganti placeholder seperti {{username}}, {{message}}, dan {{createdAt}}.

Baris kode terkait	Penjelasan
`js const messageTemplate = document.querySelector("#message-template").innerHTML;`	Mengambil template HTML untuk pesan.
`js Mustache.render(messageTemplate, { username: message.username, message: message.text, createdAt: moment(message.createdAt).format("H:mm") })`	Merender template menjadi elemen HTML yang siap ditampilkan.
`js Mustache.render(sidebarTemplate, { room, users })`	Menghasilkan tampilan sidebar berisi nama ruang dan daftar pengguna.

- **Moment →** Digunakan untuk memformat timestamp pesan yang dikirim server. Server mengirim createdAt dalam bentuk angka timestamp. Moment mengubah angka ini menjadi format jam yang mudah dibaca.

Baris kode terkait	Penjelasan
`js createdAt: moment(message.createdAt).format("H:mm")`	Mengubah timestamp menjadi jam.
`js moment(message.createdAt).format("H:mm")`	Dipakai untuk semua pesan teks dan pesan lokasi.

- **Qs →** Digunakan untuk mengambil nilai query string dari URL ketika klien masuk ke chat. Form di index.html mengirimkan user ke /chat.html?username=xxx&room=yyy. Qs mengekstrak kedua nilai tersebut agar bisa dipakai untuk proses join.

Baris kode terkait	Penjelasan
`js const { username, room } = Qs.parse(location.search, { ignoreQueryPrefix: true });`	Mengurai query string agar menjadi objek JavaScript.
Setelah diurai, nilai ini dikirim ke server melalui:	
`js socket.emit("join", { username, room }, ...)`	Qs memastikan data username dan room terbaca dengan benar dari URL.

4. Bukalah chat.js, dan perhatikan bahwa ada beberapa baris kode yang telah ditandai dengan komentar elements, templates, dan options. Jelaskan baris kode tersebut dan bagaimana kode tersebut berhubungan dengan file chat.html dan index.html.

Bagian	Fungsi	Terkait dengan file
Elements	Mengambil elemen HTML untuk dipakai dalam script	chat.html
Templates	Mengambil template Mustache untuk merender pesan dan sidebar	chat.html
Options	Mengambil username dan room dari URL	index.html (form pengirim), chat.html (halaman tujuan)

5. Jelaskan fungsi file messages.js dan users.js dan bagaimana baris kode pada kedua file ini terhubung dengan index.js, chat.js, dan kedua file HTML (chat.html dan index.html).

File	Fungsi	Terhubung ke	Hubungan
messages.js	Membuat objek pesan standar	index.js → chat.js → chat.html	Server membuat pesan → klien menerima → HTML menampilkan
users.js	Manajemen pengguna	index.js → chat.js → chat.html	Server mengatur user → klien menerima daftar user → HTML menampilkan
index.js	Server utama	messages.js, users.js, chat.js	Memancarkan data ke klien
chat.js	Logika klien	index.js, chat.html	Menangkap event, merender template
chat.html	Tampilan utama chat	chat.js	Menampilkan pesan dan daftar user
index.html	Halaman login pengguna	index.js, users.js	Mengirim username dan room ke server

6. Bagaimana aplikasi ini bisa mengirimkan lokasi. Jelaskan apa yang terjadi dengan disertai penjelasan baris kode.

Peristiwa	File dan baris kode
Browser mengambil koordinat	*navigator.geolocation.getCurrentPosition di chat.js*
Klien mengirim event kirimLokasi ke server	*socket.emit("kirimLokasi", coords) di chat.js*

Server menerima dan membentuk URL maps	*socket.on("kirimLokasi") di index.js*
Server broadcast event locationMessage	*io.to(user.room).emit("locationMessage")*
Klien menerima dan merender template link lokasi	*socket.on("locationMessage") di chat.js*
Link lokasi tampil di chat	*Template Mustache di chat.html*

7. Kenapa aplikasi ini dijalankan menggunakan perintah npm run dev bukan menggunakan perintah node diikuti nama file seperti pada jobsheet sebelumnya. Coba juga jalankan aplikasi menggunakan perintah npm run start, investigasi apa yang terjadi dan apa perbedaannya dengan npm run dev.

```
> source-code@1.0.0 start
> node src/index.js

Server is running on port 3000!
```

- **npm run start** → menjalankan aplikasi menggunakan perintah dasar node src/index.js tanpa fitur pemantauan perubahan file sehingga setiap perubahan kode mengharuskan server dihentikan dan dijalankan ulang secara manual. Mode ini lebih stabil dan biasanya dipakai ketika aplikasi sudah siap digunakan karena tidak ada proses otomatis yang berjalan di belakang.

```
> source-code@1.0.0 dev
> nodemon src/index.js

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/index.js`
Server is running on port 3000!
```

- **npm run dev** → menjalankan aplikasi memakai nodemon yang memantau seluruh perubahan pada file proyek dan melakukan restart server secara otomatis setiap kali ada penyimpanan kode sehingga proses pengembangan menjadi lebih cepat dan efisien. Mode ini memang dirancang khusus untuk developer agar tidak perlu menghentikan dan menjalankan ulang server secara manual saat melakukan iterasi kode.

8. Selain socket.on, fungsi socket apa lagi yang digunakan dalam aplikasi ini. Telusuri dan jelaskan pendapat anda disertai baris kode.

Fungsi Socket	Lokasi	Tujuan
socket.emit()	Klien dan server	Mengirim event ke server atau klien tertentu
socket.broadcast.emit()	Server	Mengirim event ke semua kecuali pengirim
socket.broadcast.to().emit()	Server	Broadcast ke room tanpa pengirim
socket.join()	Server	Memasukkan user ke room
io.to().emit()	Server	Broadcast ke seluruh member room
socket.on("disconnect")	Server	Menangani pemutusan koneksi

9. Jelaskan terkait real-time bidirectional event-based communication disertai penjelasan baris kode sesuai aplikasi yang anda buat.

Komponen	Penjelasan Singkat	Baris Kode Terkait
Komunikasi dua arah	Klien dan server saling mengirim dan menerima event tanpa perlu reload halaman	`socket.emit(...) dan socket.on(...) di chat.js dan index.js`
Klien mengirim event ke server	Pesan, lokasi, dan data join dikirim dari browser ke server secara real-time	`socket.emit("kirimPesan", pesan) • socket.emit("kirimLokasi", coords) • socket.emit("join", {...})`
Server menerima event dari klien	Server memproses data dan menyiapkan	`socket.on("kirimPesan", ...) • socket.on("kirimLokasi", ...) • socket.on("join", ...)`

	respons untuk dikirim kembali ke klien lain di room	
Server mengirim event ke klien	Server mem-broadcast pesan, lokasi, dan info room secara langsung ke semua klien di room	`io.to(user.room).emit("pesan", ...)` • `io.to(user.room).emit("locationMessage", ...)` • `io.to(user.room).emit("roomData", ...)`
Klien menerima event dari server	Klien merender pesan, link lokasi, dan pembaruan daftar anggota secara real-time	`socket.on("pesan", ...)` • `socket.on("locationMessage", ...)` • `socket.on("roomData", ...)`