BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Adapun perbandingan tinjauan pustaka yang diambil dari penetilitian sebelumnya dan penelitian yang diajukan dapat dilihat menggunakan referensi pada tabel 2.1 :

Tabel 2.1 Perbandingan Tinjauan Pustaka

Penulis	**Objek**	**Metode**	**Bahasa Pemrograman**	**Platform /Interface**
Jurista Purnama Jumri (2012).	Dosen, Mahasiswa.	SMS Gateway.	PHP, Mysql, Jquery.	Web.
M. Agung Rizkyana dan R. Sandhika Galih Amalga (2014).	Tugas, Asisten Dosen, Mahasiswa.	Node.js, Jade, Socket.io, SMS Gateway.	Javascript, MongoDB, HTML.	Web.
Timothy Yudy, Eric Anthony dan Andreas Christian (2012).	Informasi Perkuliahan.	Node.js, Websocket.	PHP, MongoDB.	Web.
Albert Leonardo Pisa, Henry Novianus Palit, dan Justinus Andjarwirawan (2016).	Audience Response, System Polling.	Socket.IO, SMS Gateway, PhoneGap.	HTML5, CSS, JavaScript, MongoDB.	Web.
M. Rizki Samsul Ariefin, Cucu Suhery, dan Yulrio Brianorman (2014).	Manajemen Mobil Antarkota	NODE.JS	Python	Web, Raspberry Pi
Nurrahman Fajar (2016)	Status Kamar.	Node.js, Socket.io.	PHP, Mysql, Javascript.	Web.

Penelitian Jurista Purnama Jumri (2012), merupakan aplikasi yang menggunakan SMS *gateway* sebagai *push notfication* secara *real-time* dengan Gammu untuk menghubungkan antara modem dengan komputer. Proses *push notification* menggunakan pulsa sehingga tidak cocok untuk aplikasi yang bersifat lokal area dengan jaringan wifi.

Penelitian M. Agung Rizkyana dan R. Sandhika Galih Amalga (2014) menggunakan node.js, socket.io dan SMS *gateway* untuk proses pengumpulan tugas dan *push notfication* antara mahasiswa dengan dosen serta asisten dosen yang dapat mengirimkan file lampiran sehingga tidak cocok untuk aplikasi yang hanya mengirimkan pesan tanpa file lampiran seperti pada penelitian Nurrahman Fajar (2016).

Penelitian Timothy Yudy, Eric Anthony dan Andreas Christian (2012) menggunakan node.js dan *websocket* untuk informasi perkuliahan antara dosen dengan mahasiswa yang dapat diakses dari mana saja. Mahasiswa hanya bisa melihat informasi saja tanpa melakukan komunikasi dengan dosen sehingga perlu ditambahkan untuk berkomunikasi dua arah.

Penelitian Albert Leonardo Pisa, Henry Novianus Palit, dan Justinus Andjarwirawan (2016) merupakan aplikasi untuk mengumpulkan data masyarakat tentang sebuah acara dengan melakukan voting secara online berbasis web dengan node.js dan socket.io tanpa adanya umpan balik atau komunikasi bersifat satu arah secara *real-time*, sehingga perlu ditambahkan untuk berkomunikasi dua arah antara masyarakat dan penyelenggara.

Penelitian M. Rizki Samsul Ariefin, Cucu Suhery, dan Yulrio Brianorman (2014) merupakan aplikasi untuk peminjaman mobil antar kota, peminjam membuka sebuah web kemudian akan mengirimkan lokasi berdasarkan Global Positioning System (GPS) ke sistem server node.js yang menggunakan Raspberry Pi kemudian sopir yang mengakses web yang sama akan mendapatkan tampilan lokasi peminjam dengan Google Maps. Aplikasi tersebut tidak cocok apabila diimplementasikan pada kasus informasi status kamar karena tidak diperlukan data lokasi.

Penelitian Nurrahman Fajar (2016) merupakan implementasi node.js dan socket.io pada kasus informasi status kamar antara FO dan HK yang bersifat lokal area menggunakan jaringan wifi dan dapat berkomunikasi dua arah secara *real-time* dengan data berupa status kamar kosong, isi, sedang perawatan dan sudah siap sewa serta menampilkan daftar kamar yang siap sewa untuk pengunjung hotel.

2.2 Dasar Teori

2.2.1 Hotel Grage Ramayana Yogyakarta

Hotel Grage Ramayana Yogyakarta adalah sebuah instansi perhotelan yang terletak di pusat Kota Yogyakarta, tepatnya di Jalan Sosrowijayan No. 33 Malioboro. Hotel Grage Ramayana Yogyakarta memiliki 72 kamar yang dilengkapi dengan berbagai macam fasilitas seperti internet gratis, resepsionis 24 jam, laundry, tv kabel, parkir, antar

jemput bandara dan lainnya untuk memberikan kenyamanan bagi para tamu menikmati kota Yogyakarta.

Hotel tersebut berdasarkan fasilitas memiliki 4 tipe yaitu, Tipe Superior yang merupakan kamar berukuran di atas kamar standar, Tipe Deluxe yang merupakan kamar diatas kamar superior yang didesain untuk terlihat lebih luas dan berkelas dalam berbagai hal, Tipe Executive yang merupakan kamar dengan ruangan lain disamping kamar tidur dan kamar mandi yaitu ruang tamu dengan beberapa kursi atau sofa. dan Tipe Family yang merupakan kamar dengan satu buah tempat tidur berukuran besar untuk dua orang dan satu buah tempat tidur berukurang satu orang. Sedangkan berdasarkan jumlah ranjang memiliki 3 tipe yaitu, Tipe Twin yang merupakan kamar untuk dua orang yang dilengkapi dengan dua buah tempat tidur masing-masing berukuran satu orang, Tipe Double yang merupakan sebuah kamar yang dilengkapi dengan satu buah tempat tidur berukuran besar untuk dua orang dan Tipe Family yang merupakan kamar dengan satu buah tempat tidur berukuran besar untuk dua orang dan satu buah tempat tidur berukuran satu orang. (dikutip dari https://www.hotelgrage.com)

2.2.2 Node.js

Node.js server merupakan platform *web server* yang dibangun menggunakan javascript dan berjalan di dalam *interpreter chrome javascript runtime*. Dibuat untuk pengembangan perangkat lunak berbasis web dengan cepat dan aplikasi jaringan yang scalable. Node.js

menggunakan *event-driven*, model *non-blocking* I/O atau bolak-balik dengan melakukan eksekusi secara *independen*, yaitu eksekusi tanpa harus menunggu eksekusi yang sebelumnya selesai dikerjakan sehingga dapat dilakukan secara bersamaan yang membuat kinerja menjadi ringan dan efisien. (dikutip dari https://nodejs.org/en/about/)

```
1  // Inisialisasi variabel dan proses
2  var waktuAwal = new Date().getTime();
3  var proses    = ()=>{
4          var waktuAkhir = new Date().getTime();
5          var lamaProses = (waktuAkhir - waktuAwal)/1000
6          return `diselesaikan dengan ${lamaProses} detik.\n`
7      }
8
9  // 3 Perintah secara urut untuk eksekusi
10 console.log('\nEksekusi Perintah Pertama', proses());
11 setTimeout(()=>{ console.log('Eksekusi Perintah Kedua', proses()) }, 1000);
12 console.log('Eksekusi Perintah Ketiga', proses());
```

Gambar 2.1 Contoh non-blocking I/O

Pada Gambar 2.1 merupakan contoh program non-blocking I/O dengan inisialisasi variabel waktu awal dan variabel proses yang merupakan sebuah fungsi, kemudia pada kode dibawahnya yaitu baris sembilan sampai dua belas merupakan tiga perintah untuk menampilkan konsol log, dengan perintah kedua merupakan perintah ekseskusi tertunda selama 1 detik. Hasil output sebagai berikut :

```
C:\Users\Seven\Desktop>node non-blocking.js
Eksekusi Perintah Pertama diselesaikan dengan 0.031 detik.
Eksekusi Perintah Ketiga diselesaikan dengan 0.031 detik.
Eksekusi Perintah Kedua diselesaikan dengan 1.045 detik.

C:\Users\Seven\Desktop>
```

Gambar 2.2 Hasil Output Contoh non-blocking I/O

Pada Gambar 2.2 terlihat bahwa hasil output eksekusi perintah kedua menmpilkan hasil paling akhir, maka terbukti bahwa eksekusi non-blocking I/O tidak harus menunggu proses sebelumnya selesai.

2.2.3 Socket.io

WebSocket adalah sebuah protokol komunikasi dua arah yang digunakan oleh browser. Dengan websocket kita tidak hanya dapat mengirimkan *request* kepada *server*, tetapi juga menerima data dari *server* tanpa harus mengirimkan *request* terlebih dahulu. Ketika menggunakan websocket akan memerlukan komponen server khusus. Salah satu komponen server untuk websocket adalah socket.io. Socket.io sendiri merupakan sebuah modul yang dapat digabungkan kedalam node.js server yang memungkinkan komunikasi dua arah secara *real-time* berbasis *event*, yaitu menerima *event* (fungsi socket.on) dan mengirim *event* (fungsi socket.emit) seperti dari *server* ke *client* dan dari *client* ke *server*.

Server (app.js)

```
var app = require('http').createServer(handler)
var io = require('socket.io')(app);
var fs = require('fs');

app.listen(80);

function handler (req, res) {
  fs.readFile(__dirname + '/index.html',
  function (err, data) {
    if (err) {
      res.writeHead(500);
      return res.end('Error loading index.html')
    }

    res.writeHead(200);
    res.end(data);
  });
}

io.on('connection', function (socket) {
  socket.emit('news', { hello: 'world' });
  socket.on('my other event', function (data) {
    console.log(data);
  });
});
```

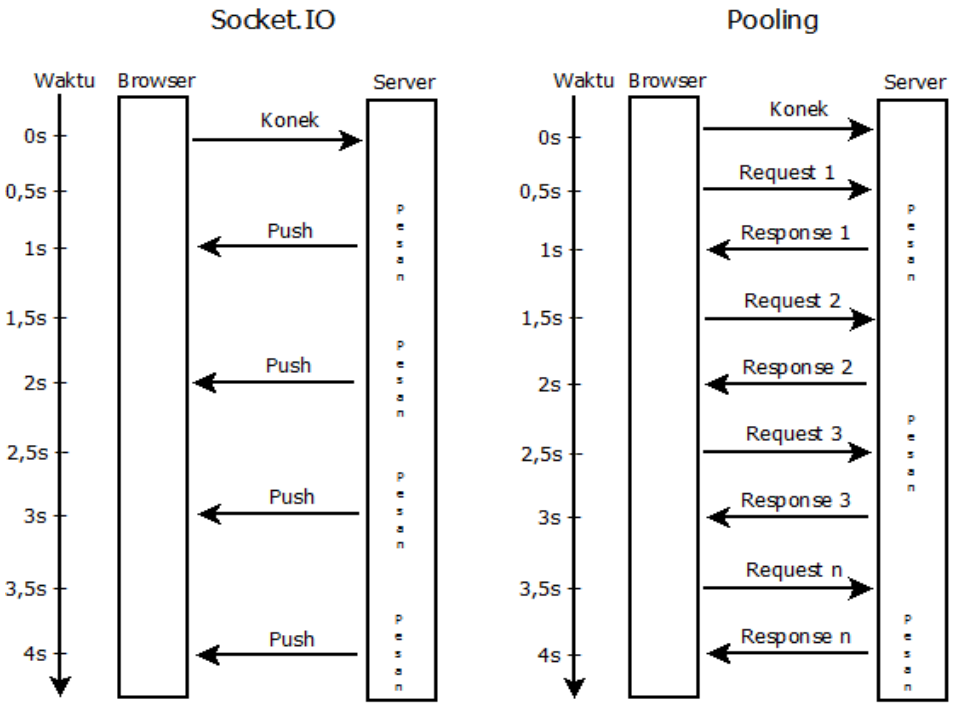
Client (index.html)

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io('http://localhost');
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'data' }
  });
</script>
```

Gambar 2.3 Contoh Penggunaan Socket.io pada Node.js Server

Pada Gambar 2.3 terlihat bahwa pada sisi server dibangun dengan menggunakan node.js dengan bahasa javascript dengan port 80 dengan fungsi handler req dan res ketika *server* diakses maka yang dijalankan adalah fungsi handler yang akan menerima *request* dan fungsi handler akan menampilkan *responses* dengan memanggil index.html serta fungsi socket.emit dan socket.on diletakkan pada io.on yang berarti socket akan memberi tanggapan ketika ada koneksi ke port 80. Pada sisi client terdapat script untuk memanggil modul socket.io.js dan fungsi socket.emit dan socket.on. (dikutip dari http://socket.io/get-started/chat/)

Keunggulan socket.io adalah tidak harus memperbarui data secara periodik perdetik atau *pooling* yang dapat memberatkan server, sehingga socket.io akan meringankan kinerja server. Dapat digambarkan sebagai berikut :



Gambar 2.4 Perbadingan Socket.io dan Pooling

Pada gambar 2.4 terlihat bahwa pada socket.io, *browser client* tidak memerlukan *request* ke *server* untuk mengecek perubahan tetapi dari *server* yang akan push ketika ada perubahan dalam periodik waktu, namun untuk pertama kali *browser client* perlu melakukan koneksi ke *server*. Sedangkan pada pooling *browser client* melakukan koneksi ke server dan secara periodik waktu memerlukan *request* ke *server* untuk mengecek perubahan dan *server* akan memberi *respon* setiap *browser client* melakukan *request*, sehingga kinerja *server* dan *browser client* menjadi berat.